9 Mnożenie macierzy a rozwiązywanie równań liniowych

9.1 Sformułowanie problemu

Zadania z tytułu są równoważne w tym sensie, że mając algorytm o złożoności $c_1 \cdot n^\alpha$ dla $\alpha \geq 2$ rozwiązujący jedno z tych zadań potrafimy rozwiązać drugie za pomocą algorytmu o złożoności $c_2 \cdot n^\alpha$ dla odpowiednio dobranej stałej c_2. Pojęcie złożoności rozumiemy tu jako złożoność algebraiczną. Tak więc algorytmem o złożoności f jest taki algorytm, który dla danych rozmiaru n wykonuje nie więcej niż $f(n)$ operacji arytmetycznych.

Oprócz wymienionych zadań będziemy także zajmować się kilkoma innymi. Wykażemy równoważność podanym sensie następujących zadań:

1) obliczania iloczynu macierzy,

2) odwracania macierzy trójkątnych,

3) odwracania dowolnych macierzy,

4) rozwiązywania układów równań liniowych.

Sformułowania zadań będą jeszcze uściślane. Będziemy też zajmować się obliczaniem tzw. rozkładu LUP.

Dowodząc równoważności będziemy często konstruować algorytmy rekurencyjne, które będzie można łatwiej opisać i analizować przy założeniu, że dane są rozmiaru będącego potęgą dwójki. Zauważmy, że podobnie jak w przypadku algorytmu mnożenia macierzy, przy pewnych założeniach algorytm o złożoności $c_1 \cdot n^\alpha$ dobrze działający dla potęg dwójki można przekształcić w algorytm rozwiązujący zadanie dla dowolnych danych bez istotnego pogarszania złożoności. Przypuśćmy, że

1) w określony sposób powiększamy dane do możliwie małego rozmiaru będącego potęgą dwójki nie wykonując żadnych działań arytmetycznych (zauważmy, że w najgorszym razie podwoimy rozmiar danych),

2) następnie uruchamiamy algorytm o złożoności $c_1 \cdot n^\alpha$ dla powiększonych danych

3) a na koniec zwracamy część obliczonych wartości (a więc też nie wykonujemy operacji arytmetycznych).

Nietrudno zauważyć, że taki algorytm ma złożoność $c_1 \cdot (2 \cdot n)^\alpha = (2^\alpha \cdot c_1) \cdot n^\alpha$.

9.2 Odwracanie macierzy trójkątnych

Zadanie 9.1 Odwracanie macierzy trójkątnej (górnej).

Dane: macierz kwadratowa A górnie trójkątna o wymiarach $n \times n$,

Rozmiar zadania: liczba wierszy macierzy A,

Obliczyć macierz odwrotną do A lub stwierdzić, że macierz odwrotna do A nie istnieje.

Zauważmy, że jeżeli istnieją macierze A^{-1} i B^{-1}, to

$$\begin{bmatrix} A & C \\ 0 & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & -A^{-1} \cdot C \cdot B^{-1} \\ 0 & B^{-1} \end{bmatrix}.$$

Stąd, w szczególności, wynika, że

$$\left[\begin{array}{cc} A & 0 \\ 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{cc} A^{-1} & 0 \\ 0 & 1 \end{array}\right]$$

dla dowolnej odwracalnej macierzy A.

Z podanych wzorów wynika następujący algorytm odwracania macierzy trójkątnych:

Algorytm $\mathcal{O}(A)$ odwracania macierzy trójkątnej (o wymiarach będących potęgą dwójki)

1) $n :=$ wymiar A;
2) jeżeli $n = 1$ i $A = [0]$, to zwróć informację, że macierz A nie jest odwracalna i zakończ działanie programu;
3) jeżeli $n = 1$ i $A = [a]$, to zwróć macierz $[a^{-1}]$ i zakończ działanie programu;
4) niech X, Y, Z będą macierzami takimi, że

$$i_{2,n/2}(A) = \left[\begin{array}{cc} X & Z \\ 0 & Y \end{array}\right]$$

(pamiętajmy, że n jest potęgą dwójki, a A jest macierzą trójkątną, stąd m.in. wynika, że X i Y są macierzami trójkątnymi);

5) $U := \mathcal{O}(X)$;
6) $V := \mathcal{O}(Y)$;
7) $W := -(U \cdot Z) \cdot V$;
8) zwróć macierz

$$i_{2,n/2}^{-1}\left(\left[\begin{array}{cc} U & W \\ 0 & V \end{array}\right]\right).$$

Przypuśćmy, że algorytm $\mathcal{O}$ korzysta z algorytmu mnożenia macierzy, który dla pewnego $\alpha \geq 2$ i dla macierzy o rozmiarze $n \times n$ wykonuje $\leq c \cdot n^\alpha$ mnożeń. Niech $O'_M(m)$ oznacza mnożeń wykonanych podczas odwracania macierzy o wymiarach $m \times m$ zgodnie z algorytmem $\mathcal{O}$. Dla argumentów będących potęgami dwójki funkcja O'_M spełnia nierówność

$$O'_M(m) \leq 2 \cdot O'_M(\frac{m}{2}) + 2 \cdot c \cdot (\frac{m}{2})^\alpha.$$

Udowodnimy, że

$$O'_M(m) \leq c \cdot m^\alpha.$$

Na podstawie zasady indukcji i założenia $\alpha \geq 2$ otrzymujemy to z następujących nierówności:

$$O'_M(m) \leq 2 \cdot c \cdot (\frac{m}{2})^\alpha + 2 \cdot c \cdot (\frac{m}{2})^\alpha \leq \frac{4}{2^\alpha} \cdot c \cdot m^\alpha \leq c \cdot m^\alpha.$$

Analogiczne oszacowania są prawdziwe dla liczby dodawań i liczby działań wykonywanych podczas realizacji algorytmu $\mathcal{O}$.

Dowolne macierze trójkątne odwracamy stosując następujący algorytm:

Algorytm odwracania macierzy trójkątnej (A jest daną macierzą)

1) Niech B oznacza możliwie małą macierz postaci

$$\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}$$

o wymiarach będących potęgą dwójki;

2) $C := \mathcal{O}(B)$;

3) nietrudno zauważyć, że macierz C jest postaci

$$\begin{bmatrix} D & 0 \\ 0 & 1 \end{bmatrix}$$

dla pewnej macierzy D o wymiarach macierzy A);

4) zwróć macierz D.

9.3 Rozkład LUP

Niech A będzie macierzą o n kolumnach i $m \leq n$ liniowo niezależnych wierszach. Pokażemy, że $A = L \cdot U \cdot P$ dla pewnych macierzy L, U i P takich, że

1) L i P są macierzami kwadratowymi,

2) L jest dolną macierzą trójkątną, tzn. jeżeli $l_{i,j}$ jest j-tym wyrazem i-tego wiersza tej macierzy i $j > i$, to $l_{i,j} = 0$,

3) wyrazy L znajdujące się na przekątnej są różne od 0 ($l_{i,i} \neq 0$),

4) U jest górną macierzą trójkątną, tzn. jeżeli $u_{i,j}$ jest j-tym wyrazem i-tego wiersza tej macierzy oraz $j < i$, to $u_{i,j} = 0$,

5) wyrazy U znajdujące się na przekątnej są różne od 0 ($u_{i,i} \neq 0$),

6) P jest macierzą permutującą, tzn. w każdym wierszu i w każdej kolumniej tej macierzy znajduje się dokładnie jeden wyraz równy 1, a pozostałe są równe 0.

Przedstawienie macierzy w postaci iloczynu macierzy o wyżej podanych własnościach nazywamy rozkładem LUP. Łatwo dowodzi się, że jeżeli macierz ma rozkład LUP, to jej wiersze są liniowo niezależne.

Zadanie 9.2 Znajdowanie rozkładu LUP.

Dane: macierz A o n kolumnach i $m = 2^l \leq n$ wierszach,

Rozmiar zadania: liczby wierszy i kolumn macierzy A,

Znaleźć rozkład LUP macierzy A lub stwierdzić, że dla macierzy A taki rozkład nie istnieje.

Algorytm $\mathcal{R}(A)$ (rozkładu macierzy, których liczba wierszy jest potęgą dwójki)

1) $m :=$ liczba wierszy A;

2) jeżeli $m = 1$ i wiersz składa się z samych zer, to zwracamy informację, że macierz A ma liniowo zależne wiersze i przerywamy wykonywanie algorytmu;

3) jeżeli $m = 1$, to ustalamy indeks i wyrazu kierunkowego macierzy (wektora) A i zwracamy $L = [1]$, P powstające z macierzy jednostkowej przez zamianę pierwszego wiersza z i-tym oraz $U = AP$ (U powstaje przez zamianę pierwszego elementu z kierunkowym, zauważmy, że PP jest macierzą jednostkową, a więc $UP = A$) i przerwamy działanie algorytmu;

4) niech B i C będą macierzami o $m/2$ wierszach takimi, że

$$A = \begin{bmatrix} B \\ C \end{bmatrix};$$

5) wywołując rekurencyjnie algorytm $\mathcal{R}$ znajdujemy L_1, U_1 i P_1 takie, że $B = L_1U_1P_1$;

6) niech $U_1 = [X, Y]$ oraz $CP_1^{-1} = [Z, T]$ dla pewnych kwadratowych macierzy X i Z.

7) obliczamy $E = ZX^{-1}$, $F = EY$ oraz $G = T - F$;

8) przedstawiamy G w postaci $L_2U_2P_2$ korzystając z rekurencyjnego wywołania algorytmu $\mathcal{R}$;

9) zwracamy

$$L = \begin{bmatrix} L_1 & 0 \\ ZX^{-1} & L_2 \end{bmatrix},$$

$$U = \begin{bmatrix} X & YP_2^{-1} \\ 0 & U_2 \end{bmatrix},$$

$$P = \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} P_1.$$

Udowodnimy teraz lematy 9.1 i 9.2, z których wynika poprawność algorytmu $\mathcal{R}(A)$. Zauważmy, że z drugiego lematu wynika także, że macierz A ma rozkład LUP wtedy i tylko wtedy, gdy jej wiersze są liniowo niezależne.

Lemat 9.1 *Jeżeli algorytm $\mathcal{R}$ zwróci macierze L, U i P, to $A = L \cdot U \cdot P$.*

Dowód. Dowodzimy ten fakt przez indukcję. Zauważmy, że

$$\begin{aligned} LUP &= \begin{bmatrix} L_1 & 0 \\ ZX^{-1} & L_2 \end{bmatrix} \begin{bmatrix} X & YP_2^{-1} \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} P_1 \\ &= \begin{bmatrix} L_1X & L_1YP_2^{-1} \\ Z & ZX^{-1}YP_2^{-1} + L_2U_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} P_1 \\ &= \begin{bmatrix} L_1X & L_1Y \\ Z & ZX^{-1}Y + L_2U_2P_2 \end{bmatrix} P_1 = \begin{bmatrix} L_1X & L_1Y \\ Z & F + G \end{bmatrix} P_1 \\ &= \begin{bmatrix} L_1X & L_1Y \\ Z & T \end{bmatrix} P_1 = \begin{bmatrix} L_1U_1 \\ CP_1^{-1} \end{bmatrix} P_1 = \begin{bmatrix} L_1U_1P_1 \\ C \end{bmatrix} = \begin{bmatrix} B \\ C \end{bmatrix} = A. \ \Box \end{aligned}$$

Lemat 9.2 *Algorytm $\mathcal{R}$ stwierdza liniową zależność wierszy macierzy A wtedy i tylko wtedy, gdy A ma liniowo zależne wiersze.*

Dowód. Najpierw ustalimy kilka pomocniczych faktów. Przypuśćmy, że mamy dwie macierze A i B. Wierszami tych macierzy są odpowiednio wektory $\vec{a}_i \in R^k$ dla $i = 1, \ldots, l$ oraz $\vec{b}_j \in R^l$. Niech $f : R^k \to R^l$ będzie funkcją zdefiniowaną wzorem

$$f(\vec{x}) = \sum_{j=1}^{l} x_j \cdot \vec{b}_j.$$

Fakt 9.3 Wektor $f(\vec{a}_i)$ jest i-tym wierszem macierzy AB. □

Fakt 9.4 Wektory $\vec{b}_1, \ldots \vec{b}_l$ są liniowo niezależne wtedy i tylko wtedy, gdy funkcja f jest wzajemnie jednoznaczna. □

Fakt 9.5 Jeżeli funkcja f jest wzajemnie jednoznaczna, to wektory $\vec{a}_1, \ldots, \vec{a}_k$ są liniowo niezależne wtedy i tylko wtedy, gdy wektory $f(\vec{a}_1), \ldots, f(\vec{a}_k)$ są liniowo niezależne. □

Fakt 9.6 Jeżeli $|A| \neq 0$, to wiersze macierzy B są liniowo niezależne wtedy i tylko wtedy, gdy wiersze macierzy AB są liniowo niezależne. Jeżeli $|\,B\,| \neq 0$, to wiersze macierzy A są liniowo niezależne wtedy i tylko wtedy, gdy wiersze macierzy AB są liniowo niezależne. □

Możemy wrócić do dowodu lematu. Będzie to dowód indukcyjny. Najpierw załóżmy, że wiersze danej macierzy A są liniowo niezależne. Wtedy macierze B i C (patrz opis algorytmu $\mathcal{R}$) są liniowo niezależne. Z założenia indukcyjnego wynika, że algorytm $\mathcal{R}$ nie stwierdzi liniowej zależności wierszy macierzy B i zwróci jej rozkład.

Będziemy teraz porównywać liniową niezależność wierszy macierzy A i G przy założeniu, że macierz $B = L_1U_1P_1$ ma wiersze liniowo niezależne. Zauważmy, że

$$\begin{bmatrix} L_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X & Y \\ Z & T \end{bmatrix} P_1 = \begin{bmatrix} L_1X & L_1Y \\ Z & T \end{bmatrix} P_1 = \begin{bmatrix} L_1U_1 \\ CP_1^{-1} \end{bmatrix} P_1 = \begin{bmatrix} B \\ C \end{bmatrix} = A.$$

Stąd i z faktu 9.6 wynika, że macierz A ma liniowo niezależne wiersze wtedy i tylko wtedy, gdy ta własność przysługuje macierzy

$$\begin{bmatrix} X & Y \\ Z & T \end{bmatrix}.$$

Mamy także

$$\begin{bmatrix} -ZX^{-1} & 1 \end{bmatrix} \begin{bmatrix} X & Y \\ Z & T \end{bmatrix} = \begin{bmatrix} 0 & T - ZX^{-1}Y \end{bmatrix} = \begin{bmatrix} 0 & G \end{bmatrix}.$$

Pierwsza z powyższych macierzy ma liniowo niezależne wiersze (ponieważ zawiera kwadratową macierz jednostkową 1). Z podanych równości otrzymujemy, że macierz A ma liniowo niezależne wiersze wtedy i tylko wtedy, gdy macierz $[0\ G]$ ma wiersze liniowo niezależne, a to jest oczywiście równoważne liniowej niezależności wierszy macierzy G.

Wobec tego i na podstawie założenia indukcyjnego drugie rekurencyjne wywołanie algorymu $\mathcal{R}$ (z macierzą G jako daną) także nie spowoduje stwierdzenia liniowej zależności wierszy macierzy A.

Dla dowodu implikacji odwrotnej załóżmy, że macierz A ma liniowo zależne wiersze. Możemy założyć, że wiersze macierzy B są liniowo niezależne, gdyż w przeciwnym razie pierwsze wywołanie rekurencyjne algorytmu $\mathcal{R}$ spowoduje stwierdzenie liniowej zależności wierszy macierzy A. Z dodatkowego założenia, stosując równoważność wykazaną w dowodzie poprzedniej implikacji pokazujemy, że także wiersze macierzy G są liniowo zależne. W tym przypadku drugie wywołanie rekurencyjne algorytmu $\mathcal{R}$ stwierdzi liniową zależność wierszy macierzy A. $\Box$

Oszacujemy złożoność algorytmu $\mathcal{R}$. Niech $R(n,m)$ oznacza liczbę operacji arytmetycznych wykonywanych podczas rozkładania macierzy o m wierszach i n kolumnach za pomocą algorytmu $\mathcal{R}$ (m powinno więc być potęgą dwójki). Analizując algorytm stwierdzamy, że $R(n,1)=0$ oraz

$$R(n,m) \le R(n,\frac{m}{2})+O(\frac{m}{2})+M(\frac{m}{2})+\left\lceil\frac{2\cdot n-m}{m}\right\rceil\cdot M(\frac{m}{2})+\frac{m}{2}(n-\frac{m}{2})+R(n-\frac{m}{2},\frac{m}{2}),$$

gdzie $O(m)$ i $M(m)$ oznaczają odpowiednio liczby operacji wymaganych do odwrócenia macierzy trójkątnej o rozmiarze $m\times m$ i do pomnożenia dwóch takich macierzy (współczynnik przy drugiej wartości funkcji M to liczba wywołań procedury mnożenia macierzy kwadratowych potrzebna do obliczenia iloczynu EY).

Przypuśćmy, że podczas wykonywania algorytmu $\mathcal{R}$ stosujemy algorytmy odwracające macierze trójkątne i mnożące macierze kwadratowe o złożoności $c\cdot m^\alpha$ dla pewnego $\alpha>2$. Weźmy liczbę

$$\gamma=\max(2,\frac{4}{2^\alpha-4}).$$

Zauważmy, że

$$\frac{2\gamma+2}{2^{\alpha-1}}\le\gamma.$$

Przez indukcję udowodnimy, że $R(n,m)\le\gamma\cdot c\cdot m^{\alpha-1}\cdot n$. Nierówność ta wynika z następujących rachunków:

$$\begin{aligned}
&R(n,m)\\
&\le R(n,\frac{m}{2})+O(\frac{m}{2})+M(\frac{m}{2})+\left\lceil\frac{2\cdot n-m}{m}\right\rceil\cdot M(\frac{m}{2})+\frac{m}{2}(n-\frac{m}{2})+R(n-\frac{m}{2},\frac{m}{2})\\
&\le\gamma\cdot c\cdot(\frac{m}{2})^{\alpha-1}\cdot n+2\cdot c\cdot(\frac{m}{2})^\alpha+\frac{2\cdot n}{m}\cdot c\cdot(\frac{m}{2})^\alpha+\frac{m}{2}\cdot n+\gamma\cdot c\cdot(\frac{m}{2})^{\alpha-1}\cdot(n-\frac{m}{2})\\
&\le c\cdot(\frac{m}{2})^{\alpha-1}\cdot n\cdot(\gamma+1+1+\gamma)+2\cdot c\cdot(\frac{m}{2})^\alpha-\gamma\cdot c\cdot(\frac{m}{2})^\alpha\\
&\le c\cdot m^{\alpha-1}\cdot n\cdot\frac{2\cdot\gamma+2}{2^{\alpha-1}}\le\gamma\cdot c\cdot m^{\alpha-1}\cdot n.
\end{aligned}$$

9.4 Zastosowania rozkładu LUP: obliczanie wyznaczników

Algorytm obliczający wyznacznik:

1) daną macierz A zastępujemy macierzą

$$B=\begin{bmatrix}A&0\\0&1\end{bmatrix}$$

o wymiarze będącym potęgą dwójki (zauważmy, że macierze A i B mają te same wyznaczniki);

2) znajdujemy przedstawienie macierzy B w postaci LUP (uruchamiamy algorytm rozwiązujący zadanie 9.2),

3) jeżeli macierz B została przedstawiona w postaci LUP, to obliczamy wyznaczniki macierzy L, U oraz P, mnożymy je przez siebie i zwracamy obliczony iloczyn, w przeciwnym razie zwracamy 0.

Ponieważ macierze L i U są trójkątne, więc ich wyznaczniki obliczamy mnożąc elementy z przekątnych, macierz P ma wyznacznik równy 1 lub -1 i można go znaleźć ustalając liczbę przestawień niezbędną do przekształcenia P w macierz jednostkową.

9.5 Zastosowania rozkładu LUP: odwracanie macierzy

Zadanie 9.3 Odwracanie macierzy.

Dane: macierz kwadratowa A,

Rozmiar zadania: liczba wierszy macierzy A,

Obliczyć macierz A^{-1} lub stwierdzić, że macierz odwrotna do A nie istnieje.

Algorytm obliczający macierz odwrotną:

1) daną macierz A zastępujemy macierzą

$$B = \begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}$$

o wymiarze będącym potęgą dwójki,

2) obliczamy przedstawienie macierzy B w postaci LUP uruchamiając algorytm rozwiązujący zadanie 9.2,

3) jeżeli macierz B ma przedstawienie postaci LUP, to znajdujemy B^{-1} korzystając ze wzoru $B^{-1} = P^{-1}U^{-1}L^{-1}$, w przeciwnym razie stwierdzamy, że macierz B nie jest odwracalna.

Zauważmy, że

$$\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & 1 \end{bmatrix}.$$

Wzór $(AB)^T = B^T A^T$ implikuje, że $A^{-1} = ((A^T)^{-1})^T$ i pozwala obliczyć macierz odwrotną do macierzy dolnej trójkątnej stosując algorytm obliczania macierzy odwrotnej do górnej macierzy trójkątnej. Ponadto $P^{-1} = P^T$. Oczywiście, złożoność tego algorytmu jest tego samego rzędu, co złożoność algorytmu mnożenia macierzy.

9.6 Mnożenie macierzy za pomocą odwracania

Wzór

$$\begin{bmatrix} 1 & A & 0 \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -A & AB \\ 0 & 1 & -B \\ 0 & 0 & 1 \end{bmatrix}$$

pozwala zredukować mnożenie macierzy do obliczania macierzy odwrotnej. Jeżeli posłużymy się algorytmem odwracającym macierz o złożoności $c \cdot n^\alpha$, to oparty o podany wzór algorytm obliczający iloczyn macierzy będzie miał złożoność $(c \cdot 3^\alpha) \cdot n^\alpha$.

Przedstawiony rezultat dowodzi, że równoważne są wszystkie zadania wymienione w rozdziale 9.1 z wyjątkiem ostatniego (rozwiązywania równań liniowych). Równoważność ostatniego zadania i pozostałych wynika z następnego rozdziału.

9.7 Rozwiązywanie układów równań liniowych

Ustalmy najpierw czym jest rozwiązanie układu równań liniowych. Aby rozwiązać układ $A\vec{x} = \vec{b}$ można znaleźć jedno rozwiązanie tego układu (jeżeli istnieje) i bazę (lub zbiór generatorów) przestrzeni $\{\vec{x} : A\vec{x} = \vec{0}\}$.

Rozważmy układ równań liniowych postaci $\vec{x} = A\vec{y} + \vec{b}$, gdzie A jest macierzą o m wierszach długości n. Z postaci tego układu, bez żadnych obliczeń, możemy odczytać jego rozwiązanie. Tak więc wektor $(\vec{x}, \vec{y})$ równy $(\vec{b}, \vec{0}) \in R^{n+m}$ spełnia to równanie, a przestrzeń $\{(\vec{x}, \vec{y}) \in R^{n+m} : \vec{x} = A\vec{y}\}$ jest generowana przez wektory $(\vec{a}_i, \vec{e}_i)$, gdzie $\vec{a}_i$ jest i-tą kolumną macierzy A, dla $1 \leq i \leq n$. Albo inaczej, przestrzeń ta jest generowana przez kolumny macierzy $\begin{bmatrix} A \\ 1 \end{bmatrix}$. Nietrudno też zauważyć, że opisany zbiór generatorów składa się z wektorów liniowo niezależnych. Wobec tego, zadanie rozwiązania danego układu równań może być rozumiane jako polecenie znalezienia równoważnego mu układu postaci $\vec{x} = A\vec{y} + \vec{b}$, a sam ten układ możemy uważać za rozwiązanie danego.

Będzie interesować nas kilka zadań. Najbardziej szczegółowym jest następujące.

Zadanie 9.4 Rozwiązać układ równań liniowych wyrażając określone zmienne przez pozostałe.

> Dane: układ $A\vec{x} = \vec{b}$ złożony z $m \leq n$ równań liniowych z n niewiadomymi,
>
> Rozmiar: liczba równań m i liczba niewiadomych n,
>
> Znaleźć macierz C i wektor $\vec{d}$ takie, że dany układ równań jest równoważny układowi postaci $(x_1, \ldots, x_m) = C(x_{m+1}, \ldots, x_n) + (d_1, \ldots, d_m)$, gdzie C oznacza tu pewną macierz o m wierszach i $n-m$ kolumnach (wektory utożsamiamy z jednokolumnowymi macierzami) lub stwierdzić, że nie jest to możliwe (albo mówiąc inaczej: wyrazić pierwsze m zmiennych przez pozostałe lub stwierdzić, że rozwiązania nie da się tak wyrazić).

Zadanie 9.4 można rozwiązać stosując następujący algorytm:

Algorytm rozwiązujący zadanie 9.4

1) Macierz A przedstawiamy w postaci $[A_1, A_2]$, gdzie A_1 jest macierzą kwadratową,
2) obliczamy macierz odwrotną do macierzy A_1 uruchamiając algorytm rozwiązujący zadanie 9.3,
3) jeżeli A_1 ma macierz odwrotną, to obliczamy i zwracamy iloczyny $C = A_1^{-1}A_2$ i $\vec{d} = A_1^{-1}\vec{b}$, w przeciwnym razie stwierdzamy, że dany układ nie ma rozwiązania żądanej postaci.

Nietrudno zauważyć, że złożoność tego algorytmu wyraża się wzorem $c \cdot m^{\alpha-1} \cdot n$, gdzie α jest wykładnikiem z wzoru szacującego złożoność algorytmu mnożenia macierzy. Co więcej, algorytm ten świadczy o tym, zadanie 9.4 można zredudować do problemu odwracania macierzy (zadanie 9.3). Wątpliwości może jednak budzić jego poprawność. Wynika ona z następującego lematu:

Lemat 9.7 *Jeżeli A jest macierzą kwadratową, a układy $A\vec{x} + B\vec{y} = \vec{c}$ oraz $\vec{x} + D\vec{y} = \vec{e}$ są równoważne, to A jest macierzą odwracalną.*

Dowód. Przypuśćmy, że $A\vec{a} = \vec{0}$ (a więc rozważamy liniową kombinację kolumn macierzy A ze współczynnikami $\vec{a}$, równą zero). Każde równanie takiej postaci, jak drugie, ma rozwiązanie. Wobec tego, rozwiązanie musi mieć też pierwsze równanie. Weźmy dowolne jego rozwiązanie $(\vec{x}_0, \vec{y}_0)$. Rozwiązaniem pierwszego równania jest też $(\vec{x}_0 + \vec{a}, \vec{y}_0)$. Korzystając z równoważności obu równań, otrzymujemy, że rozwiązania te spełniają drugie równanie. Tak więc

$$\vec{x}_0 + \vec{a} + D\vec{y}_0 = \vec{e} = \vec{x}_0 + D\vec{y}_0.$$

Stąd wynika, że $\vec{a} = \vec{0}$. W ten sposób dowiedliśmy, że kolumny macierzy A są liniowo niezależne, a to implikuje tezę. □

Zauważmy, że prawdziwy jest też następujący lemat:

Lemat 9.8 *Jeżeli układy* $\vec{x} + B_1\vec{y} = \vec{c}_1$ *oraz* $\vec{x} + B_2\vec{y} = \vec{c}_2$ *są równoważne, to* $B_1 = B_2$ *i* $\vec{c}_1 = \vec{c}_2$.

Dowód. Oczywiście, wektor $(\vec{c}_1, \vec{0})$ jest rozwiązaniem pierwszego równania, a więc jest także rozwiązaniem drugiego. Podstawiając ten wektor do drugiego równania otrzymujemy, że $\vec{c}_1 = \vec{c}_2$. Z tej równości z kolei wynika, że $B_1\vec{y} = B_2\vec{y}$ dla wszystkich $\vec{y}$. Tak więc $B_1\vec{e}_i = B_2\vec{e}_i$ dla wszystkich dopuszczalnych i. Równość ta oznacza, że i-te kolumny macierzy B_1 i B_2 są identyczne. □

Z lematu 9.8 wynika następujący algorytm odwracania macierzy (zadanie 9.3):

Algorytm odwracania macierzy

1) Dla danej macierzy A tworzymy układ równań $A\vec{x} - I\vec{y} = \vec{0}$, gdzie I jest macierzą jednostkową o takich samych wymiarach, jak macierz A,

2) obliczamy rozwiązanie równania $A\vec{x} - I\vec{y} = \vec{0}$ postaci $\vec{x} - B\vec{y} = \vec{c}$,

3) Jeżeli znajdziemy takie rozwiązanie, to zwracamy macierz B, w przeciwnym razie stwierdzamy, że macierz A nie jest odwracalna.

Z lematu 9.7 wynika, że powyższy algorytm uruchomiony z macierzą nieodwracalną stwierdzi ten fakt, a jeżeli uruchomimy go z macierzą odwracalną A, to znajdzie rozwiązanie równania $A\vec{x} - I\vec{y} = \vec{0}$ i obliczy odpowiednią macierz B. Mnożąc to równanie stronami przez A^{-1} otrzymujemy $\vec{x} - A^{-1}\vec{y} = \vec{0}$. Z lematu 9.8 wynika, że $B = A^{-1}$, a więc w tym przypadku algorytm zwróci macierz odwrotną do A. Zauważmy także, że dwa ostatnie algorytmy świadczą o równoważności zadań 9.4 i 9.3.

Polecenie rozwiązania układu równań liniowych może też być rozumiane jako zadanie 9.5 i rozwiązane w inny sposób.

Zadanie 9.5 Rozwiązywanie układu niezależnych równań liniowych

Dane: układ $A\vec{x} = \vec{b}$ złożony z m liniowo niezależnych równań (liniowych) z n niewiadomymi (tak więc $m \leq n$),

Rozmiar: liczba równań i liczba niewiadomych,

Znaleźć rozwiązanie danego układu równań.

Algorytm rozwiązujący zadanie 9.5

1) dany układ m równań z n zmiennymi powiększamy do układu złożonego 2^l równań dopisując możliwie mało równań postaci $y_i = 0$, używając nowych zmiennych y_i; przypuśćmy, że w ten sposób otrzymaliśmy równanie $A'\vec{x}' = \vec{b}'$

2) macierz A' danego układu o μ wierszach długości ν przedstawiamy w postaci LUP stosując algorytm $\mathcal{R}$,

3) rozwiązujemy układ $L\vec{y} = \vec{b}'$ stosując metodę macierzy równoważnych wierszowo; niech $\vec{y}_0$ będzie znalezionym rozwiązaniem,

4) obliczamy $\vec{z} = P\vec{x}'$ (czyli przestawiamy odpowiednio zmienne w ciągu $\vec{x}'$),

5) rozwiązujemy układ $U\vec{z} = \vec{y}_0$ stosując algorytm rozwiązujący zadanie 9.4 i wyrażając pierwsze m zmiennych ciągu $\vec{z}$ przez pozostałe,

6) zwracamy rozwiązanie znalezione w poprzednim punkcie usuwając z niego informacje o dodatkowych zmiennych y_i.

Zauważmy, że rozwiązanie metodę macierzy równoważnych wierszowo układu $L\vec{y} = \vec{b}$ takiego, że A jest macierzą trójkątną (górną lub dolną) o wymiarach $\mu \times \mu$ wymaga wykonania najwyżej $c \cdot \mu^2$ operacji arytmetycznych. Przestawienie zmiennych w ciągu $\vec{x}'$ (obliczenie $P\vec{x}'$ dla permutacyjnej macierzy P) nie wymaga żadnych operacji arytmetycznych. Stosując odpowiednie algorytmy rozkładania macierzy i rozwiązywania zadania 9.4, punkty 1 i 4 rozważanego algorytmy można zrealizować wykonującą $c \cdot \mu^{\alpha-1} \cdot \nu$ operacji arytmetycznych. Wobec tego podany algorytm wykonuje najwyżej

$$c \cdot \mu^{\alpha-1} \cdot \nu + c_1 \cdot \mu^2 + c \cdot \mu^{\alpha-1} \cdot \nu \leq (2 \cdot c + c_1) \cdot \mu^{\alpha-1} \cdot \nu$$

operacji arytmetycznych. Ponieważ $\mu < 2 \cdot m$ i $\nu < 2 \cdot n$, więc analizowany algorytm ma złożoność

$$\leq (2 \cdot c + c_1) \cdot 2^{\alpha} \cdot m^{\alpha-1} \cdot n.$$

Polecenie rozwiązania układu równań liniowych najbardziej ogólnie może być rozumiane jako

Zadanie 9.6 Rozwiązywanie układu równań liniowych.

Dane: układ $A\vec{x} = \vec{b}$ złożony z m równań liniowych z n niewiadomymi,

Rozmiar: liczba równań i liczba niewiadomych,

Znaleźć rozwiązanie danego układu równań lub stwierdzić, że układ nie ma rozwiązania.

Zadanie 9.6 może zostać zredukowane do zadania 9.5. Można to zrobić w sposób opisany w kolejnych rozdziałach.

10 Baza przestrzeni generowanej przez dane wektory

10.1 Sformułowanie zadania, związek z rozwiązywaniem równań

Macierz A z m wierszami i n kolumnami, o wyrazach $a_{i,j}$ dla $i \leq m$ i $j \leq n$ nazywa się schodkową, jeżeli istnieje taki ciąg $j_1, \ldots, j_r$ liczb spełniających nierówności $1 \leq j_1 < \ldots < j_r \leq n$, że

1) $a_{i,j} = 0$ dla wszystkich $i = 1, \ldots, r$ i dla $j < j_i$,

2) $a_{i,j} = 0$ dla wszystkich $i > r$ i dla wszystkich $j = 1, \ldots, n$,

3) $a_{i,j_i} \neq 0$ dla wszystkich $i = 1, \ldots, r$.

Będziemy rozwiązywać następujące zadanie:

Zadanie 10.1 Sprowadzanie macierzy do postaci schodkowej.

Dane: macierz A o m wierszach i n kolumnach,

Rozmiar zadania: liczby m i n,

Znaleźć odwracalną macierz U o wymiarach $m \times m$ taką, że UA jest macierzą schodkową oraz iloczyn UA (często żąda się, aby dodatkowo $|\,U\,| = 1$).

Zauważmy, że wiersze macierzy UA są kombinacjami liniowymi wierszy macierzy A. Ponieważ dla odwracalnej macierzy U zachodzi także równość $A = U^{-1}(UA)$, więc w takim przypadku macierze A oraz UA generują tę samą przestrzeń. Jeżeli UA jest macierzą schodkową, to jej niezerowe wiersze są liniowo niezależne. Wobec tego, jeżeli rozwiążemy zadanie 10.1, to także znajdziemy szczególną bazę przestrzeni generowanej przez wiersze danej macierzy.

Wykorzystując zadanie 10.1 można rozwiązać zadanie 9.6 wykonując na przykład następujący algorytm:

Algorytm 10.1 znajdujący rozwiązanie układu równań liniowych

1) Przypuśćmy, że mamy dany układ $A\vec{x} = \vec{b}$ złożony z m równań z n zmiennymi. Najpierw uruchamiamy algorytm dla zadania 10.1. Przyjmijmy, że algorytm ten zwróci macierze U oraz $S = UA$,

2) Niech r będzie liczbą niezerowych wierszy macierzy S. Macierz S przedstawiamy w postaci $\begin{bmatrix} S_1 \\ 0 \end{bmatrix}$, gdzie S_1 jest macierzą o r (niezerowych, liniowo niezależnych) wierszach.

3) Obliczamy $\vec{c} = U\vec{b}$ i dzielimy go na dwie części: $\vec{c}_1$ długości r i resztę $\vec{c}_2$.

4) Sprawdzamy, czy $\vec{c}_2$ ma wszystkie wspólrzedne równe 0; jeżeli tak nie jest, to kończymy obliczenia stwierdzając, że dany układ równań jest sprzeczny i nie ma rozwiązania.

5) rozwiązujemy układ $S_1\vec{x} = \vec{c}_1$ uruchamiając np. algorytm dla zadania 9.5 i zwracamy rozwiązanie znalezione przez ten algorytm.

10.2 Zadania pomocnicze

Najpierw będziemy zajmować się kilkoma zadaniami pomocniczymi. Algorytmy rozwiązujące te zadania łatwiej sformułować przy założeniu, że dane macierze mają wymiary będące potęgami dwójki. Rozważane zadania charakteryzują się tym, że trudno je rozwiązać powiększając rozmiar danych (np. do potęgi dwójki), rozwiązując zadanie dla większych danych, a następnie przekazując część obliczonych wartości. Przedstawiając algorytmy będziemy przyjmować jednak dodatkowe założenia o rozmiarze danych. Aby rozwiązać zadanie w ogólnym przypadku wystarczy przekonać się, że dodatkowe założenia nie są istotne i w podobny sposób, komplikując opis, można algorytmy przekształcić w równie dobre, dzielące dane w przybliżeniu na pół.

Zadanie 10.2 Częściowe sprowadzanie do macierzy trójkątnych.

Dane: macierz A o $m \leq n$ wierszach i n kolumnach oraz górna trójkątna macierz R o wymiarach $n \times n$.

Rozmiar zadania: liczba n.

Znaleźć odwracalną macierz U oraz górną trójkątną macierz R' o wymiarach $n \times n$ taką, że

$$U \begin{bmatrix} A \\ R \end{bmatrix} = \begin{bmatrix} R' \\ 0 \end{bmatrix}.$$

Algorytm 10.2 rozwiązujący zadanie 10.2 (przyjmujemy dodatkowo, że n jest potęgą dwójki).

1) jeżeli $n = m = 1$, to $A = [a]$, $R = [r]$ dla pewnych a i r i wtedy przerywamy działanie algorytmu 10.2 zwracając macierz

$$U = \begin{bmatrix} 1 & 0 \\ -ra^{-1} & 1 \end{bmatrix} \text{ lub } U = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

zależnie od tego, czy $a \neq 0$, czy też $a = 0$, i odpowiednią macierz R';

2) jeżeli $m = 0$, to przerywamy działanie algorytmu 10.2 zwracając macierze $U = 1$ i $R' = R$.

3) Dalej będziemy zakładać, że jeszcze $m = n$. Jeżeli $m < n$, to postępujemy analogicznie posługując się mniejszymi macierzami. Macierz

$$\begin{bmatrix} A \\ R \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \\ R_1 & A_5 \\ 0 & R_2 \end{bmatrix}$$

dzielimy na 8 podmacierzy o wymiarach $n/2 \times n/2$ (zauważmy, że macierze R_1 i R_2 są trójkątne)

4) wywołujemy rekurencyjnie algorytm 10.2 dla macierzy A_3 i R_1, przypuśćmy, że wywołanie to zwraca macierze U_1 i R_3;

5) obliczamy

$$\begin{bmatrix} A_6 \\ A_7 \end{bmatrix} = U_1 \begin{bmatrix} A_4 \\ A_5 \end{bmatrix};$$

6) wywołujemy rekurencyjnie algorytm 10.2 dla macierzy A_1 i R_3, przypuśćmy, że wywołanie to zwraca macierz U_2 i R_4;

7) obliczamy

$$\begin{bmatrix} A_8 \\ A_9 \end{bmatrix} = U_2 \begin{bmatrix} A_2 \\ A_6 \end{bmatrix};$$

8) wywołujemy rekurencyjnie algorytm 10.2 dla macierzy A_7 i R_2, przypuśćmy, że wywołanie to zwraca macierze U_3 i R_5;

9) wywołujemy rekurencyjnie algorytm 10.2 dla macierzy A_9 i R_5, przypuśćmy, że wywołanie to zwraca macierz U_4 i R_6;

10) obliczamy i zwracamy (1 to macierz jednostkowa o wymiarach $n/2 \times n/2$)

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & U_3 \end{bmatrix} \begin{bmatrix} U_2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

11) zwracamy

$$R' = \begin{bmatrix} R_4 & A_8 \\ 0 & R_6 \end{bmatrix}.$$

Poprawność algorytmu 10.2 wynika z następujących rachunków:

$$U\begin{bmatrix} A \\ R \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & U_3 \end{bmatrix}\begin{bmatrix} U_2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & U_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \\ R_1 & A_5 \\ 0 & R_2 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & U_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & U_3 \end{bmatrix}\begin{bmatrix} U_2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} A_1 & A_2 \\ R_3 & A_6 \\ 0 & A_7 \\ 0 & R_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & U_3 \end{bmatrix}\begin{bmatrix} R_4 & A_8 \\ 0 & A_9 \\ 0 & A_7 \\ 0 & R_2 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & U_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} R_4 & A_8 \\ 0 & A_9 \\ 0 & R_5 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R_4 & A_8 \\ 0 & R_6 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R' \\ 0 \end{bmatrix}.$$

Niech α będzie większe od 2 i niech $M(n) = c_1 \cdot n^\alpha$ będzie złożonością algorytmu mnożenia macierzy o wymiarach $n \times n$. Jeżeli $f(n)$ oznacza liczbę operacji arytmetycznych wykonywanych przez wyżej algorytm 10.2, to $f(1) = 1$ oraz

$$f(n) \leq 4 \cdot f(\frac{n}{2}) + c_2 \cdot M(\frac{n}{2}) = 4 \cdot f(\frac{n}{2}) + c \cdot n^\alpha$$

dla pewnej stałej c. Z tej nierówności otrzymujemy, że

$$f(n) \leq \frac{2^\alpha \cdot c}{2^\alpha - 4} \cdot (n^\alpha - n^2) + n^2$$

dla wszystkich n będących potęgą dwójki i – w konsekwencji – $f(n) \leq c_3 \cdot M(n)$. Pierwszą z dwóch ostatnich nierówności dowodzimy przez indukcję w następujący sposób:

$$f(n) \leq 4 \cdot \left[\frac{2^\alpha \cdot c}{2^\alpha - 4} \cdot ((\frac{n}{2})^\alpha - (\frac{n}{2})^2) + (\frac{n}{2})^2\right] + c \cdot n^\alpha =$$
$$\frac{4 \cdot c}{2^\alpha - 4} \cdot n^\alpha - \frac{2^\alpha \cdot c}{2^\alpha - 4} \cdot n^2 + \frac{(2^\alpha - 4) \cdot c}{2^\alpha - 4} \cdot n^\alpha + n^2 = \frac{2^\alpha \cdot c}{2^\alpha - 4} \cdot (n^\alpha - n^2) + n^2.$$

Zadanie 10.3 Sprowadzanie do macierzy trójkątnych.

Dane: macierz kwadratowa A o n wierszach.

Rozmiar zadania: liczba n.

Znaleźć odwracalną macierz U oraz górną trójkątną macierz R o wymiarach $n \times n$ taką, że $UA = R$.

Algorytm 10.3 rozwiązujący zadanie 10.3 (dodatkowo zakładamy, że n jest potęgą dwójki).

1) jeżeli $n = 1$, to $A = [a]$ dla pewnego a i wtedy przerywamy działanie algorytmu 10.3 zwracając macierze $U = [1]$ i A.

2) macierz A dzielimy na cztery podmacierze

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix};$$

3) wywołujemy rekurencyjnie algorytm 10.3 dla macierzy A_3, przypuśćmy, że wywołanie to zwraca macierze U_1 i R_1;

4) obliczamy $A_5 = U_1 A_4$;

5) wywołujemy algorytm 10.2 dla macierzy A_1 i R_1, przypuśćmy, że wywołanie to zwraca macierz U_2 i R_2;

6) obliczamy

$$\begin{bmatrix} A_6 \\ A_7 \end{bmatrix} = U_2 \begin{bmatrix} A_2 \\ A_5 \end{bmatrix};$$

7) wywołujemy rekurencyjnie algorytm 10.3 dla macierzy A_7, przypuśćmy, że wywołanie to zwraca macierze U_3 i R_3;

8) obliczamy i zwracamy (1 to macierz jednostkowa o wymiarach $n/2 \times n/2$)

$$U = \begin{bmatrix} 1 & 0 \\ 0 & U_3 \end{bmatrix} U_2 \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix}.$$

9) zwracamy

$$R' = \begin{bmatrix} R_2 & A_6 \\ 0 & R_3 \end{bmatrix}.$$

Poprawność algorytmu 10.3 wynika z następujących rachunków:

$$UA = \begin{bmatrix} 1 & 0 \\ 0 & U_3 \end{bmatrix} U_2 \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix} \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & U_3 \end{bmatrix} U_2 \begin{bmatrix} A_1 & A_2 \\ R_1 & A_5 \end{bmatrix} =$$
$$\begin{bmatrix} 1 & 0 \\ 0 & U_3 \end{bmatrix} \begin{bmatrix} R_2 & A_6 \\ 0 & A_7 \end{bmatrix} = \begin{bmatrix} R_2 & A_6 \\ 0 & R_3 \end{bmatrix} = R.$$

Jeżeli teraz $f_2(n)$ oznacza liczbę operacji arytmetycznych wykonywanych przez algorytm 10.3, to $f_2(1) = 0$ oraz

$$f_2(n) \leq 2 \cdot f_2(\frac{n}{2}) + f_1(\frac{n}{2}) + c_2 \cdot M(\frac{n}{2}) \leq 2 \cdot f_2(\frac{n}{2}) + c \cdot n^\alpha$$

dla pewnej stałej c. Z tej nierówności otrzymujemy, że

$$f_2(n) \leq \frac{2^\alpha \cdot c}{2^\alpha - 2} \cdot (n^\alpha - n)$$

dla wszystkich n będących potęgą dwójki i – w konsekwencji – $f_2(n) \leq c_4 \cdot M(n)$. Pierwszą z tych nierówności dowodzimy przez indukcję tak, jak w poprzednim przypadku:

$$f_2(n) \leq 2 \cdot \frac{2^\alpha \cdot c}{2^\alpha - 2} \cdot ((\frac{n}{2})^\alpha - \frac{n}{2}) + c \cdot n^\alpha =$$
$$\frac{2 \cdot c}{2^\alpha - 2} \cdot n^\alpha - \frac{2^\alpha \cdot c}{2^\alpha - 2} \cdot n + \frac{(2^\alpha - 2) \cdot c}{2^\alpha - 2} \cdot n^\alpha = \frac{2^\alpha \cdot c}{2^\alpha - 2} \cdot (n^\alpha - n).$$

Zadanie 10.4 Sprowadzanie macierzy trójkątnej do macierzy schodkowej.

Dane: kwadratowa trójkątna górna macierz R o n wierszach.

Rozmiar zadania: liczba n.

Znaleźć odwracalną macierz U oraz schodkowa macierz S o wymiarach $n \times n$ taką, że $UR = S$.

Algorytm 10.4 rozwiązujący zadanie 10.4 (zakładamy, że n jest potęgą dwójki).

1) jeżeli $n = 1$, to $R = [a]$ dla pewnego a i wtedy przerywamy działanie algorytmu 10.4 zwracając macierze $U = [1]$ i R.

2) macierz R dzielimy na cztery podmacierze
$$R = \begin{bmatrix} R_1 & A_1 \\ 0 & R_2 \end{bmatrix};$$

3) wywołujemy rekurencyjnie algorytm 10.4 dla macierzy R_1, przypuśćmy, że wywołanie to zwraca macierze U_1 i macierz schodkową złożoną z macierzy S_1 o niezerowych wierszach i kilku wierszy zerowych;

4) obliczamy $\begin{bmatrix} A_2 \\ A_3 \end{bmatrix} = U_1 A_1$, przy czym A_2 ma tyle samo wierszy, co S_1;

5) wywołujemy algorytm 10.2 dla macierzy A_3 i R_2, przypuśćmy, że wywołanie to zwraca macierz U_2 i R_3;

6) wywołujemy rekurencyjnie algorytm 10.4 dla macierzy R_3, przypuśćmy, że wywołanie to zwraca macierze U_3 i S_2;

7) obliczamy i zwracamy
$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} U_1 & 0 \\ 0 & 1 \end{bmatrix}.$$

8) zwracamy
$$S = \begin{bmatrix} S_1 & A_2 \\ 0 & S_2 \\ 0 & 0 \end{bmatrix}.$$

Poprawność algorytmu 10.4 wynika z następujących rachunków:
$$UR = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} U_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_1 & A_1 \\ 0 & R_2 \end{bmatrix} =$$
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & U_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & A_2 \\ 0 & A_3 \\ 0 & R_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & U_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_1 & A_2 \\ 0 & R_3 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} S_1 & A_2 \\ 0 & S_2 \\ 0 & 0 \end{bmatrix} = S.$$

Jeżeli teraz $f_3(n)$ oznacza liczbę operacji arytmetycznych wykonywanych przez algorytm 10.4, to $f_3(1) = 0$ oraz
$$f_3(n) \leq 2 \cdot f_3(\frac{n}{2}) + f_1(\frac{n}{2}) + c_5 \cdot M(\frac{n}{2}) \leq 2 \cdot f_3(\frac{n}{2}) + c \cdot n^\alpha$$
dla pewnej stałej c. Z tej nierówności, podobnie jak w poprzednim przypadku, otrzymujemy, że
$$f_2(n) \leq \frac{2^\alpha \cdot c}{2^\alpha - 2} \cdot (n^\alpha - n)$$
dla wszystkich n będących potęgą dwójki i – w konsekwencji – $f_2(n) \leq c_6 \cdot M(n)$.

10.3 Sprowadzanie macierzy do postaci schodkowej

Zadanie 10.5 Sprowadzanie do postaci schodkowej macierzy o krótkich wierszach.

Dane: macierz A o m wierszach i $n \leq m$ kolumnach,

Rozmiar zadania: liczba m,

Znaleźć odwracalną macierz U o wymiarach $m \times m$ oraz macierz schodkową S taką, że $UA = S$.

Algorytm 10.5 rozwiązujący zadanie 10.5.

1) Tworzymy kwadratową macierz $A_1 = [A, 0]$;

2) uruchamiamy dla macierzy A_1 algorytm 10.3; przypuśćmy, że algorytm ten zwrócił macierze U_1 i R;

3) uruchamiamy dla macierzy R algorytm 10.4; przypuśćmy, że algorytm ten zwrócił macierze U_2 i S;

4) Znajdujemy macierz S_1 składającą się z n pierwszych kolumn macierzy S i zwracamy macierze $U = U_1 U_2$ oraz S_1.

Zadanie 10.6 Sprowadzanie dowolnej macierzy do postaci schodkowej.

Dane: macierz A o m wierszach i n kolumnach,

Rozmiar zadania: liczby m i n,

Znaleźć odwracalną macierz U o wymiarach $m \times m$ oraz macierz schodkową S taką, że $UA = S$.

Algorytm 10.6 rozwiązujący zadanie 10.6.

1) $Y := A$, $U := 1$, $k := m$, X i S są macierzami pustymi (zmienna k pamięta liczba wierszy macierzy A, która ma być przekształcana, cała przekształcana macierz jest pamiętana jako $[S \;\; X \;\; Y]$, macierz S zawiera część już przekształconą i jest schodkowa, macierze X i Y zawierają przekształconą w różnym stopniu resztę),

2) dopóki k ostatnich wierszy macierzy $[X \; Y]$ zawiera niezerowy element

   (a) jeżeli X ma mniej niż k kolumn, to

      (a.1) $l := \min\{m, \text{liczba kolumn macierzy } Y\}$,

      (a.2) dzielimy macierz Y na dwie części: macierz Y_1 o l kolumnach i resztę Y_2 (tak więc $Y = [Y_1 \; Y_2]$);

      (a.3) $X := [X \;\; UY_1]$ oraz $Y := Y_2$

   (b) $p := \min\{k, \text{liczba kolumn macierzy } X\}$,

   (c) dzielimy macierz X na dwie części: macierz X_1 o p kolumnach i resztę X_2 (tak więc $X = [X_1 \; X_2]$);

   (d) $X := X_2$,

   (e) przedstawiamy macierz X_1 w postaci $\begin{bmatrix} X_3 \\ X_4 \end{bmatrix}$, gdzie X_4 ma k wierszy,

   (f) jeżeli macierz X_4 jest zerowa, to $S := [S \;\; X_1]$ (i zaczynamy wykonywać pętlę dopóki kolejny raz od początku), a w przeciwnym razie

(g) uruchamiamy dla macierzy X_4 algorytm 10.5; przypuśćmy, że algorytm ten zwrócił U_2 i macierz schodkową $S_1 = \begin{bmatrix} S_2 \\ 0 \end{bmatrix}$, gdzie S_2 jest macierzą o niezerowych wierszach;

(h) $X := \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} X$ oraz $U := \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} U$,

(i) $k := k -$ liczba wierszy macierzy S_2,

(j) $S := \left[S \begin{bmatrix} X_3 \\ S_2 \\ 0 \end{bmatrix} \right]$

3) zwracamy macierze U oraz $[S \;\; X \;\; UY]$.

Nietrudno zauważyć, że suma ilości kolumn macierzy S, X i Y jest niezmiennikiem pętli dopóki w algorytmie 10.6 i jest równa liczbie kolumn danej macierzy A. Jeżeli zmienna k ma wartość 0, to warunek sterujący pętlą staje się fałszywy i algorytm 10.6 zatrzymuje się. Jeżeli zaś na początku pętli $k > 0$, to także zmienna p przyjmuje wartość dodatnią. Ponieważ podczas każdego wykonania pętli zwiększa się o p liczba kolumn macierzy S, więc algorytm 10.6 kończy pracę dla dowolnych danych.

Przypuśćmy, że $k_1, k_2, \ldots, k_t$ są wartościami zmiennej k bezpośrednio po rozpoczęciu wykonywania kolejnych pętli, a $p_1, p_2, \ldots, p_t$ – odpowiadającymi im wartościami zmiennej p. Oczywiście, $p_i \leq k_i \leq m$ dla $i \leq t$. Jeżeli tuż po zdefiniowaniu wartości p będzie prawdziwa nierówność $p_i < k_i$, to liczba kolumn macierzy $[X_2 \;\; Y]$ jest równa 0, czyli macierz $[X_2 \;\; Y]$ jest pusta. W tej sytuacji algorytm 10.6 wykonuje pętlę po raz ostatni. Wobec tego, $p_i = k_i$ dla wszystkich $i < t$. Z poprzedniego akapitu wynika, że po zakończeniu działania algorytmu 10.6 macierz S ma $\sum_{i=1}^{t} p_i$ kolumn. W ten sposób otrzymujemy, że $\sum_{i=1}^{t} p_i \leq n$. Wobec tego prawdziwa jest także nierówność $\sum_{i=1}^{t} k_i \leq n + m$.

Łatwo zauważyć, że stwierdzenie „k ostatnich wierszy macierzy S składa się z samych zer” jest niezmiennikiem pętli w algorytmie 10.6 i – w konsekwencji – stwierdzenie to jest prawdziwe na początku i na końcu każdej pętli.

Przyjmijmy, że wyrażenie $c \cdot m^{\alpha}$ szacuje z góry liczbę operacji arytmetycznych wykonywanych zarówno podczas mnożenia macierzy kwadratowych o m wierszach, jak i podczas realizacji algorytmu 10.5 dla macierzy o takiej liczbie wierszy. Zauważmy także, że instrukcja wykonywana w pętli dopóki ma postać

$$\text{if war}_1 \text{ then instr}_1\text{; instr}_2\text{; if war}_2 \text{ then instr}_3 \text{ else instr}_4.$$

Podczas wykonania algorytmu 10.6 operacje arytmetyczne są wykonywane tylko podczas wykonywania instrukcji instr_1 lub instr_4, oraz po wykonaniu pętli dopóki. Zauważmy, że obliczając iloczyn

$$\begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} U,$$

gdzie macierz U_2 ma wymiary $k \times k$, a macierz U – $m \times m$, możemy ostatnie k wierszy macierzy U podzielić na macierze o wymiarach $k \times k$ i każdą otrzymaną w ten sposób macierz pomnożyć przez U_2. Wtedy cały iloczyn składa się z $m - k$ pierwszych wierszy macierzy U i obliczonych iloczynów o wymiarach $k \times k$. Podobnie możemy obliczać drugi iloczyn z punktu (h). Wobec tego, podczas wykonywania pojedyńczej instrukcji instr_4 jest realizowane

$$c \cdot k^{\alpha} + c \cdot \left\lceil \frac{m}{k} \right\rceil k^{\alpha} + c \cdot \left\lceil \frac{m}{k} \right\rceil k^{\alpha} \leq (3 \cdot c + 2) \cdot m^{\alpha - 1} \cdot k$$

operacji arytmetycznych (pamiętajmy, że $k \leq m$). Podczas całego algorytmu 10.6, wykonanie instrukcji instr_4 wymaga

$$\sum_{i=1}^{t}(3\cdot c+2)\cdot m^{\alpha-1}\cdot k_i \leq (3\cdot c+2)\cdot m^{\alpha-1}\cdot \sum_{i=1}^{t} k_i \leq (3\cdot c+2)\cdot m^{\alpha-1}\cdot(n+m)\leq$$
$$\leq (6\cdot c+4)\cdot m^{\alpha-1}\cdot \max(n,m)$$

operacji arytmetycznych.

Zrealizowanie wszystkich instrukcji instr_1 i obliczenia kończące wymagają łącznie wykonania $\lceil \frac{n}{m} \rceil$ mnożeń macierzy kwadratowych o wymiarach $m\times m$. Uwzględniając fakt, że za pierwszym razem mnożymy przez macierz $U = 1$, te fragmenty algorytmu wymagają wykonania $c\cdot m^{\alpha-1}\cdot n$ operacji arytmetycznych. Przeprowadzone obliczenia dowodzą, że algorytm 10.6 uruchomiony z macierzą o m wierszach i n kolumnach wykonuje najwyżej

$$(7\cdot c+4)\cdot m^{\alpha-1}\cdot \max(n,m)$$

operacji arytmetycznych.

Pokażemy jeszcze, że równość $UA = [S\ \ X\ \ UY]$ jest niezmiennikiem pętli algorytmu 10.6. Oczywiście, równość ta jest prawdziwa przed pierwszym wykonaniem pętli. Jeżeli jest niezmiennikiem, jest też prawdziwa po wykonaniu pętli. Poprawność algorytmu 10.6 jest konsekwencją tego faktu i spostrzeżenia, że macierz $[S\ \ X\ \ UY]$ jest schodkowa. Tak jest, gdyż S jest schodkowa i k jej ostatnich wierszy składa się z samych zer, a z warunku zakończenia pętli wynika, że tę samą własność ma k ostatnich wierszy macierzy $[X\ \ Y]$, oraz – w konsekwencji – macierzy $[X\ \ UY]$.

Aby dowieść, że podana równość jest niezmiennikiem przyjmijmy, że indeks p oznacza początkową wartość zmiennej, a indeks k – końcową. Mamy więc

$$U_pA = [S_p\ \ X_p\ \ U_pY_p] = [S_p\ \ X_p\ \ U_p[Y_1\ \ Y_2]] = [S_p\ \ X_p\ \ U_pY_1\ \ U_pY_2] =$$
$$= [S_p\ \ [X_p\ \ U_pY_1]\ \ U_pY_2] = [S_p\ \ [X_1\ \ X_2]\ \ U_pY_k] = [S_p\ \ \begin{bmatrix} X_3 \\ X_4 \end{bmatrix}\ \ X_2\ \ U_pY_k].$$

Jeżeli powyższą równość pomnożymy przez macierz $U_3 = \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix}$ i skorzystamy z definicji U_k, to otrzymamy

$$U_kA = [U_3S_p\ \ \begin{bmatrix} X_3 \\ U_2X_4 \end{bmatrix}\ \ U_3X_2\ \ U_3U_pY_k] = [S_p\ \ \begin{bmatrix} X_3 \\ S_2 \\ 0 \end{bmatrix}\ \ X_k\ \ U_kY_k] = [S_k\ \ X_k\ \ U_kY_k].$$

Z powyższych rozważań wynika

Twierdzenie 10.1 (Keller-Gehrig) *Zadanie 10.6 można rozwiązać dla danej macierzy A o m wierszach i n kolumnach wykonując $c\cdot m^{\alpha-1}\cdot \max(n,m)$ operacji arytmetycznych, gdzie c jest pewną stałą, a α jest wykładnikiem z oszacowania złożoności algorytmu mnożenia macierzy.* □

Wniosek 10.2 *Dowolny układ m równań liniowych z n niewiadomymi można rozwiązać (można rozwiązać zadanie 9.6) wykonując $c\cdot m^{\alpha-1}\cdot \max(n,m)$ operacji arytmetycznych, gdzie c jest pewną stałą, a α jest wykładnikiem z oszacowania złożoności algorytmu mnożenia macierzy.*

Dowód. Wniosek wynika z analizy algorytmu 10.1, który w odpowiednich momentach uruchamia algorytm 10.6 i algorytm o odpowiedniej złożoności rozwiązujący zadanie 9.5 (np. algorytm przedstawiony w rozdz. 4). □