Algorytmika
Ćwiczenia 7

June 10, 2025

Adrian Herda
Informatyka Algorytmiczna
Politechnika Wrocławska

1. Zadanie 33 - wyznacznik Vandermonda

1.1. Treść

Pokazać że

$$V(x_1, ..., x_n) = \prod_{i<j} (x_j - x_i)$$

1.2. Rozwiązanie

Macierz Vandermonda jest zdefiniowana jako:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ 1 & x_2 & x_2^2 & ... & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^n \end{pmatrix}$$

A jej wyznacznik to

$$V(x_1, ..., x_n) = \begin{vmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ 1 & x_2 & x_2^2 & ... & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^n \end{vmatrix}$$

Główna idea dowodu: Jeśli do kolumny macierzy dodamy (lub od niej odejmiemy) inną kolumnę pomnożoną przez pewien skalar, to wyznacznik macierzy nie zmienia się.

A więc w każdej kolumnie oprócz pierwszej odejmujemy poprzednią pomnożoną przez x_0. To daje nam macierz:

$$V = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 1 & x_1 - x_0 & x_1(x_1 - x_0) & ... & x_1^{n-1}(x_1 - x_0) \\ 1 & x_2 - x_0 & x_2(x_2 - x_0) & ... & x_2^{n-1}(x_2 - x_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & x_n(x_n - x_0) & ... & x_n^{n-1}(x_n - x_0) \end{pmatrix}$$

Teraz wykonując rozwinięcie Laplace'a względem pierwszego wiersza otrzymujemy $\det(V) = \det(B)$ gdzie:

$$B = \begin{pmatrix} x_1 - x_0 & x_1(x_1 - x_0) & \dots & x_1^{n-1}(x_1 - x_0) \\ x_2 - x_0 & x_2(x_2 - x_0) & \dots & x_2^{n-1}(x_2 - x_0) \\ \vdots & \vdots & \ddots & \vdots \\ x_n - x_0 & x_n(x_n - x_0) & \dots & x_n^{n-1}(x_n - x_0) \end{pmatrix}$$

Jako że wszystkie wartości na i-tym wierszu mają współczynnik w postaci $x_{i+1} - x_0$ możemy je wyciągnąć przed macierz i otrzymać równość:

$$\begin{aligned} \det(V) &= (x_1 - x_0)(x_2 - x_0)\cdots(x_n - x_0) \begin{vmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{vmatrix} \\ &= \prod_{1 \le i \le n} (x_i - x_0) \det(V') \end{aligned}$$

gdzie V' jest macierzą Vandermonda dla $x_1, ..., x_n$. powtarzając ten proces na coraz mniejszych macierzach Vandermonda otrzymujemy produkt:

$$\begin{aligned} \det(V) &= \prod_{0<i\le n} (x_i - x_0) \cdot \prod_{1<i\le n} (x_i - x_1) \cdot \dots \cdot \prod_{n-1<i\le n} (x_i - x_{n-1}) \\ &= \prod_{0\le j<n} \left(\prod_{j<i\le n} (x_i - x_j) \right) \\ &= \prod_{0\le j<i\le n} (x_i - x_j) && \text{Zamieniając notacje } i - j \\ &= \prod_{0\le i<j\le n} (x_j - x_i) \end{aligned}$$

■

2. Zadanie 36 - maksymalne sparowanie dzięki specyfikacji

2.1. Treść

Należy udowodnić że jeśli w grafie $G = (V, E)$ zachodzą warunki specyfikacji S albo *single* to zbiór $M = \{(p, \mathrm{pref}_p) : \mathrm{pref}_p \neq \mathrm{NULL}\}$ jest sparowaniem maksymalnym.

$$S = (\forall_{p \in V})(\mathrm{married}(p) \lor \mathrm{single}(p))$$

2.2. Dowód

Dowód będzie nie wprost

Załóżmy przeciwnie, że zbiór M nie jest maksymalnym sparowaniem, to znaczy że istnieje M' taki że $M \subset M'$.

1. M jest poprawnym sparowaniem. Z definicji *married*:

$$\mathrm{married}(p) \equiv \mathrm{pref}_p = q \in N(p) \land \mathrm{pref}_q = p \in N(q)$$

Zatem jeśli $\text{pref}_q \neq \text{NULL}$ to żeby dodać (p, pref_p) do M, musimy mieć:
- $\text{pref}_p = q$
- $\text{pref}_q = p$

A więc jako że każdy wierzchołek może być albo *married* albo *single* to żaden wierzchołek nie może być częścią obu par.

2. Załóżmy że M nie jest maksymalne

   Jako że M nie jest maksymalne to znaczy że istnieje para $(p, q) \in M' \setminus M$ którą można by dodać do M. Oznacza to, że:
   - $\text{pref}_p = \text{NULL}$,
   - $\text{pref}_q = \text{NULL}$,
   - $p \in N(q) \land q \in N(p)$,

   To oznacza że oba wierzchołki są *free* co zaprzecza warunkom specyfikacji.

Zatem M musi być sparowaniem maksymalnym.

■

3. Zadanie 37 - specyfikacja kończy jakąkolwiek pracę w algorytmie

3.1. Treść

Należy udowodnić że jeśli zachodzą warunki specyfikacji S to konfiguracja jest ostateczna (żaden krok algorytmu nie zmieni konfiguracji)

$$S = (\forall_{p \in V})(\text{married}(p) \lor \text{single}(p))$$

3.1.1. Algorytm

```
do forever
  if pref_p == NULL && (exists q in N(p))(pref_q == p)
    pref_p ← q
  end if
  if pref_p == NULL
    && (forall q in N(p))(pref_q != p)
    && (exists q in N(p))(pref_q == NULL)
    pref_p ← q
  end if
  if pref_p == q && pref_q != p && pref_q != NULL
    pref_p ← NULL
  end if
end do
```

3.2. Dowód

Jeśli zachodzą warunki specyfikacji S to znaczy że nie ma żadnych wierzchołków w stanach *free*, *wait* oraz *chain*.

1. Pierwsza klauzula `if` sprawdza czy istnieje wierzchołek p taki że $\mathrm{pref}_p = \mathrm{NULL}$ oraz istnieje wierzchołek $q \in N(p)$ taki że $\mathrm{pref}_q = p$ a to znaczyłoby że wierzchołek q musi być w stanie *wait*. Jako że nie ma już takich wierzchołków dzięki warunkom specyfikacji, ta klauzula nie może zostać wykonana.
2. Druga klauzula `if` sprawdza istnienie wierzchołka p takiego że $\mathrm{pref}_p = \mathrm{NULL}$ oraz takiego wierzchołka $q \in N(p)$ że $\mathrm{pref}_q = \mathrm{NULL}$. Taka klauzula spełniona byłaby tylko gdyby $\mathrm{free}(p) \wedge \mathrm{free}(q)$. Jako że nie ma już takich wierzchołków dzięki warunkom specyfikacji, ta klauzula nie może zostać wykonana.
3. Trzecia klauzula `if` sprawdza istnienie wierzchołka p takiego że $\mathrm{pref}_p = q$ oraz takiego wierzchołka $q \in N(p)$ że $\mathrm{pref}_q \neq p \wedge \mathrm{pref}_q \neq \mathrm{NULL}$ to znaczy że $\mathrm{pref}_q = r \wedge r \neq p$. To znaczyłoby że raka klauzula byłaby spełniona tylko dla wierzchołka $\mathrm{chain}(p)$. Jako że nie ma już takich wierzchołków dzięki warunkom specyfikacji, ta klauzula nie może zostać wykonana.

Z punktów powyższych klauzul wynika że algorytm nie może wykonać żadnego kroku, a więc konfiguracja jest ostateczna.

■