Metoda Dziel i Zwyciężaj (cd.)

IIUWr. II rok informatyki. *Opracował: Krzysztof Loryś*

1 Dalsze przykłady

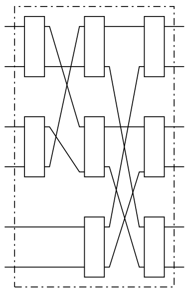
1.1 Sieci przełączników

Przełącznikiem dwustanowym nazywamy urządzenie o dwóch portach wejściowych i dwóch portach wyjściowych, które:

- w stanie 1 przesyła dane z wejścia i na wyjście i (dla $i = 0, 1$),
- w stanie 2 przesyła dane z wejścia i na wyjście $(i + 1) \bmod 2$ (dla $i = 1, 2$).

Rysunek 1: *Przełącznik dwustanowy*

Łącząc ze sobą przełączniki otrzymujemy *sieci przełączników*, które poprzez różne ustawienia przełączników mogą realizować różne permutacje danych.



Rysunek 2: *Przykład sieci przełączników o 6 wejściach*

Problem:
Dane: liczba naturalna n.
Zadanie: skonstruować sieć $Perm_n$ przełączników realizującą wszystkie permutacje n elementów.

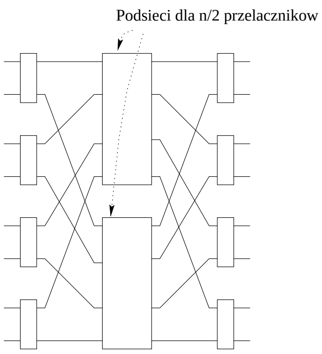
Kryteriami określającymi jakość sieci są:

- liczba przełączników;
- głębokość sieci, tj. długość maksymalnej drogi od portu wejściowego do portu wyjściowego. Długość ta mierzona jest liczbą przełączników znajdujących się na drodze (oczywiście połączenia między przełącznikami są jednokierunkowe).

1.1.1 Konstrukcja

Dla prostoty ograniczymy się do konstrukcji sieci dla n będącego potęgą liczby 2.

Konstrukcja oparta jest za zasadzie Dziel i Zwyciężaj i sprowadza się do sprytnego rozesłania danych wejściowych do dwóch (zbudowanych rekurencyjnie) egzemplarzy sieci o $n/2$ wejściach, a następnie na umiejętnym połączeniu portów wyjściowych tych podsieci. Istota tej konstrukcji przedstawiona jest na poniższym rysunku.



Rysunek 3: *Konstrukcja sieci dla* $n = 8$

Porty wejściowe sieci połączone przełącznikiem w pierwszej warstwie oraz porty wyjściowe połączone przełącznikiem w ostatniej warstwie będziemy nazywać portami *sąsiednimi*.

Fakt 1 *Tak skonstruowana sieć ma głębokość* $2 \log n - 1$ *i zawiera* $\Theta(n \log n)$ *przełączników.*

Dowód: Głębokość sieci wyraża się równaniem

$$G(2^k) = \begin{cases} 1 & \text{dla } k = 1 \\ G(2^{k-1}) + 2 & \text{dla } k > 1 \end{cases}$$

a liczba przełączników - równaniem:

$$P(2^k) = \begin{cases} 1 & \text{dla } k = 1 \\ 2P(2^{k-1}) + 2^k & \text{dla } k > 1 \end{cases}$$

□

1.1.2 Poprawność konstrukcji

Niech π będzie dowolną permutacją n-elementową. Pokażemy, że istnieje ustawienie przełączników sieci realizujące π, tj. takie, że dane z i-tego portu sieci zostaną przesłane na $\pi(i)$-ty port wyjściowy (dla $i = 1, \ldots, n$). Istnienie takiego ustawienia będzie konsekwencją istnienia odpowiedniego dwukolorowania wierzchołków w następującym grafie $G_\pi = (V, E)$.

- Zbiór $V = V_I \cup V_O \cup V_M$ składa się z:
  - n wierzchołków (podzbiór V_I) odpowiadających portom wejściowym sieci;
  - n wierzchołków (podzbiór V_O) odpowiadających przełącznikom z ostatniej warstwy sieci (po dwa wierzchołki na każdy przełącznik);
  - n wierzchołków (podzbiór V_M) dodanych ze względów technicznych.
- Wszystkie wierzchołki z V etykietujemy:
  - wierzchołek z V_I odpowiadający i-temu portowi wejściowemu otrzymuje etykietę i;

- wierzchołki z V_O, z pary odpowiadającej j-temu przełącznikowi ostatniej warstwy, otrzymują etykiety i" i k", takie, że $2j-1=\pi(i)$ oraz $2j=\pi(k)$ (innymi słowy na porty wyjściowe j-tego przełącznika mają być wysłane wartości z i-tego oraz k-tego portu wejściowego sieci);
- wierzchołki z V_M otrzymują w dowolny sposób różne etykiety ze zbioru $\{1', 2', \ldots, n'\}$.

- Zbiór $E = E_I \cup E_O \cup E_M$ składa się z:
  - $n/2$ krawędzi (podzbiór E_I) łączących wierzchołki o etykietach $2i-1$ i $2i$, a więc takie, które odpowiadają sąsiednim portom wejściowym;
  - $n/2$ krawędzi (podzbiór E_O) łączących wierzchołki odpowiadające temu samemu przełącznikowi ostatniej warstwy;
  - $2n$ krawędzi (podzbiór E_M) łączących wierzchołki o etykietach i i i' oraz wierzchołki o etykietach i' i i".

Fakt 2 *Graf G_π jest sumą rozłącznych cykli parzystej długości.*

DOWÓD: Stopień każdego wierzchołka w G_π jest równy 2. □

Z faktu tego wprost wynika istnienie kolorowania wierzchołków G_π dwoma kolorami (powiedzmy białym i czarnym). Kolorowanie to ma następujące własności:

- Wierzchołki odpowiadające sąsiednim portom (zarówno wejściowym jak i wyjściowym) otrzymują różne kolory.
- Wierzchołki o etykietach i oraz i" otrzymują ten sam kolor (dla każdego $i = 1, \ldots, n$).

Stąd wnioskujemy istnienie ustawienia przełączników realizującego π:

- Przełączniki z pierwszej warstwy ustawiamy tak, by dane z portów białych (dokładniej: których odpowiadające wierzchołki otrzymały kolor biały) były przesłane do górnej podsieci $Perm_{n/2}$.
- Przełączniki w górnej podsieci $Perm_{n/2}$ ustawiamy tak, by permutowała swoje dane zgodnie z permutacją π. Dokładniej:
  Niech $K = k_1, \ldots, k_{n/2}$ będzie ciągiem etykiet białych wierzchołków z V_I w kolejności ich występowania w ciągu $\{1, 2, \ldots, n\}$ a $L = l_1, \ldots, l_{n/2}$ ciągiem etykiet białych wierzchołków z V_O w kolejności ich występowania w ciągu $\pi(1), \ldots, \pi(n)$. Niech $\pi_a : \{1, \ldots, n/2\} \to \{1, \ldots, n/2\}$ będzie permutacją taką, że $\pi_a(i) = j$ wtedy i tylko wtedy gdy $l_j = k_i$". Przełączniki ustawiamy tak, by podsieć realizowała permutację π_a. Takie ustawienie istnieje na mocy indukcji. Podobne rozważania przeprowadzamy dla dolnej podsieci $Perm_{n/2}$.
- Dla przelączników ostatniej warstwy stosujemy następującą regułę:
  Niech i" będzie etykietą białego wierzchołka z pary odpowiadającej j-temu przełącznikowi, a k" - etykietą czarnego wierzchołka z tej pary. Jeśli i poprzedza k w permutacji π (tzn. $i = \pi(2j-1)$ oraz $k = \pi(2j)$) przełącznik ustawiamy na wprost, w przeciwnym razie ustawiamy na ukos.

1.2 Para najbliżej położonych punktów

PROBLEM:
Dane: Zbiór $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ współrzędnych punktów na płaszczyźnie.
Zadanie: Znaleźć dwa najbliżej położone względem siebie punkty w P, tj. znaleźć i,j takie, że $d(x_i, y_i, x_j, y_j) = min\{d(x_k, y_k, x_l, y_l) \mid 1 \le k < l \le n\}$, gdzie $d(x_k, y_k, x_l, y_l) = \sqrt{(x_k - x_l)^2 + (y_k - y_l)^2}$.

Siłowe rozwiązanie, polegające na wyliczeniu i porównaniu odległości między każdą parą punktów, wymaga czasu $\Omega(n^2)$. Przedstawimy teraz strategię Dziel i Zwyciężaj, która daje algorytm działający w czasie $\Theta(n \log n)$.

Terminologia: mówiąc o punkcie r będziemy mieć na myśli punkt o współrzędnych (x_r, y_r).

1.2.1 Strategia Dziel i Zwyciężaj

1. (a) Sortujemy punkty z P według współrzędnych x i zapamiętujemy je w tablicy X;
   (b) Sortujemy punkty z P według współrzędnych y i zapamiętujemy je w tablicy Y;
   (c) Znajdujemy prostą l dzielącą P na dwa równoliczne (z dokładnością do 1) podzbiory:
      - P_L- podzbiór punktów leżących na lewo od l,
      - P_R - podzbiór punktów leżących na prawo od l.

      Punkty znajdujące się na prostej l (o ile są takie) kwalifikujemy do tych podzbiorów w dowolny sposób.

2. { rekurencyjnie }
   $(i_1, j_1) \leftarrow$ para punktów z P_L o najmniejszej odległości;
   $(i_2, j_2) \leftarrow$ para punktów z P_R o najmniejszej odległości.

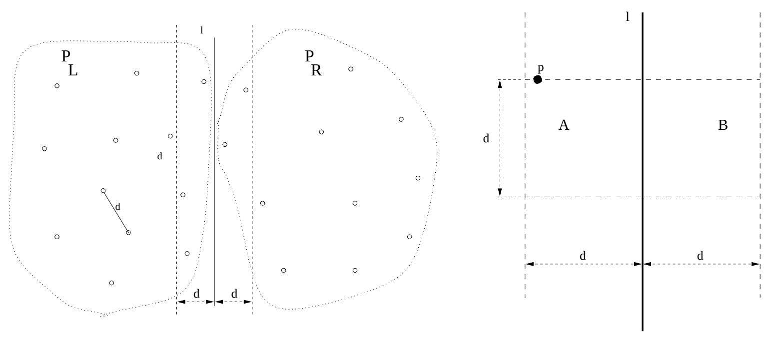
3. Niech (i', j') będzie tą parą punktów znalezioną w kroku 2, dla której odległość (oznaczmy ją przez d) jest mniejsza.
   Sprawdzamy czy istnieje para punktów (t, s) odległych o mniej niż d takich, że $t \in P_L$ i $s \in P_R$. Jeśli istnieje, przekazujemy ją jako wynik procedury, w przeciwnym razie jako wynik przekazujemy parę (i', j').

$\Box$

Wyjaśnienia wymaga sposób realizacji kroku 3. Oznaczmy przez P_C zbiór tych punktów z P, które leżą w odległości nie większej niż d od prostej l. Niech Y' oznacza tablicę Y, z której usunięto wszystkie punkty spoza P_C. Korzystamy z następującego spostrzeżenia:

Fakt 3 *Jeśli (t, s) jest parą punktów odległych o mniej niż d taką, że $t \in P_L$ i $s \in P_R$, to t i s należą do P_C. Ponadto w tablicy w Y' pomiędzy t a s leży nie więcej niż 6 punktów.*

Dowód: Gdyby jeden z punktów leżał w odległości większej niż d od prostej l, to odległość między nimi byłaby większa niż d. Oczywiste jest też, że współrzędne y-kowe tych punktów różnią się nie więcej niż o d. Tak więc punkty t i s leżą w prostokącie o wymiarach $d \times 2d$ jak pokazano na rysunku 4.



Rysunek 4: *(a) W kroku 3 pary (t, s) należy szukać tylko w zaznaczonym pasie (b) Jeśli p ma być jednym z punktów pary (t, s), to drugi punkt musi znajdować się w kwadracie A. Ponadto wszystkie punkty z Y' leżące między t a s muszą leżeć w A lub w B.*

W części A leżą tylko punkty z P_L. Ponieważ każde dwa z nich odległe są od siebie o co najmniej d, więc może ich tam znajdować się co najwyżej 4. Z analogicznego powodu w części B może znajdować się nie więcej niż 4 punkty z P_R. Tak więc w całym prostokącie znajduje się nie więcej niż 8 punktów. $\square$

Krok 3 sprowadza się więc do utworzenia tablicy Y', a następnie do obliczenia odległości każdego punktu z Y' do co najwyżej siedmiu punktów następujących po nim w tej tablicy.

1.2.2 Koszt:

Krok 1:

- Sortowanie - $\Theta(n \log n)$.
- Znalezienie prostej l i podział P na podzbiory - koszt stały.

Krok 2: $2T(n/2)$

Krok 3:

- Utworzenie Y' - $\Theta(n)$.
- Szukanie pary (t, s) - $\Theta(n)$.

Stąd koszt całego algorytmu wyraża się równaniem $T(n) = 2T(n/2)+\Theta(n \log n)$, którego rozwiązaniem jest $\Theta(n \log^2 n)$. Koszt ten można zredukować do $\Theta(n \log n)$. Wystarczy zauważyć , że sortowanie punktów w każdym wywołaniu rekurencyjnym jest zbyteczne. Zbiór P możemy przekazywać kolejnemu wywołaniu rekurencyjnemu jako tablice X i Y. Na ich podstawie można w czasie liniowym utworzyć odpowiednie tablice dla zbiorów P_L i P_R. Tak więc sortowanie wystarczy przeprowadzić jeden raz - przed pierwszym wywołaniem procedury rekurencyjnej.

Po takiej modyfikacji czas wykonania procedury rekurencyjnej wyraża się równaniem $T(n) = 2T(n/2) + \Theta(n)$, którego rozwiązaniem jest $\Theta(n \log n)$. Dodany do tego czas sortowania nie zwiększa rzędu funkcji.