Многопоточное программирование

Параллельные алгоритмы на графах

Граф

- $G = (V, E)$
- Направленные и не направленные
- Дерево
- Цикличные и ацикличные
- $G = (V, E, w)$

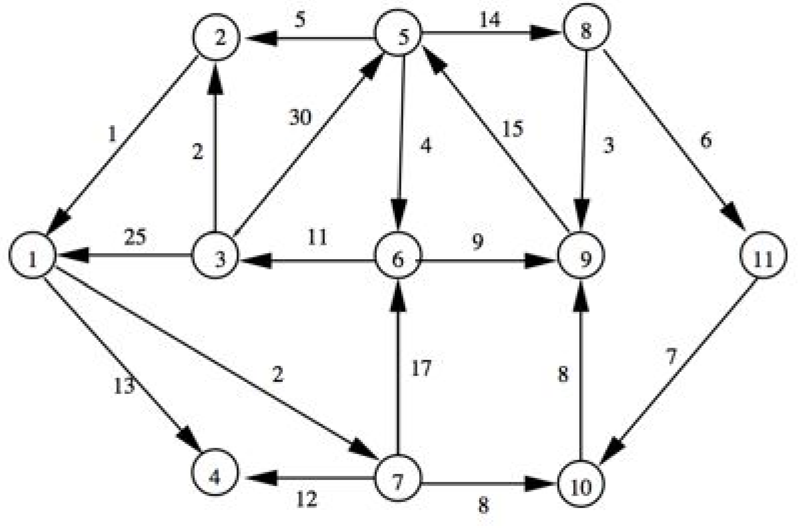
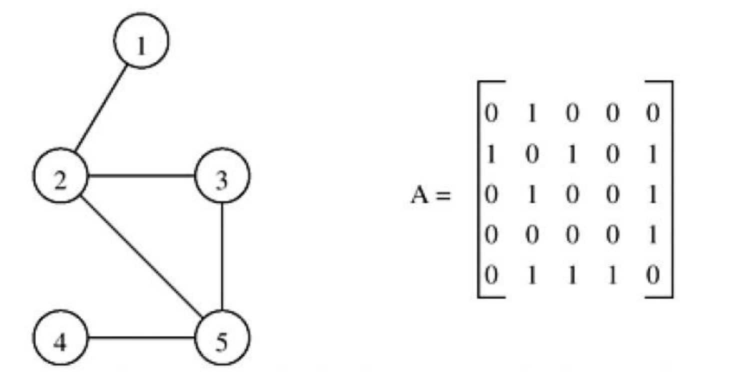


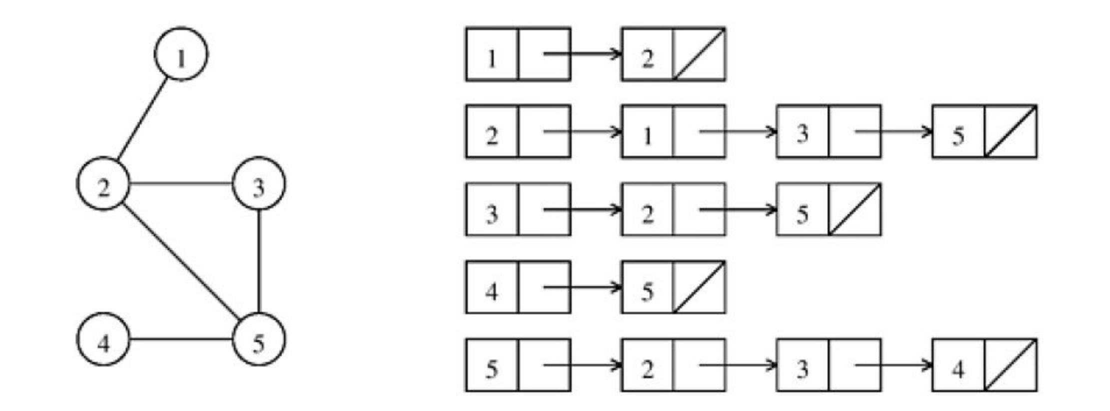
Figure 4

Граф

1. Матрица смежности

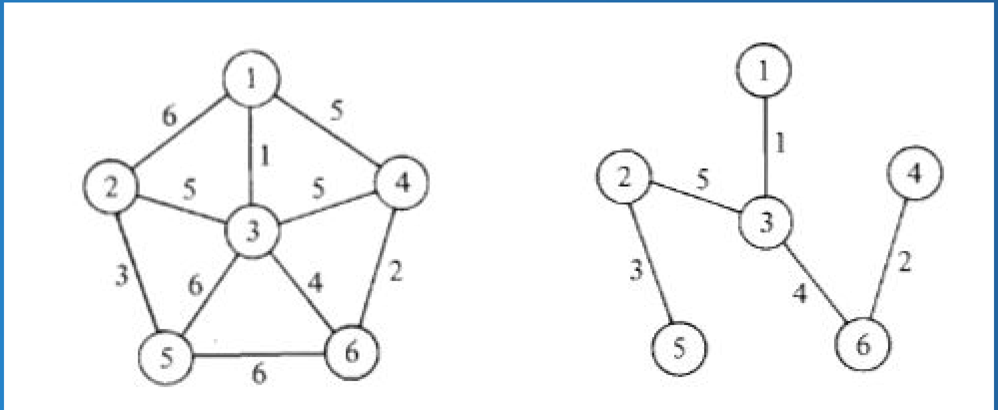


2. Списки связности



Минимальное остовное дерево

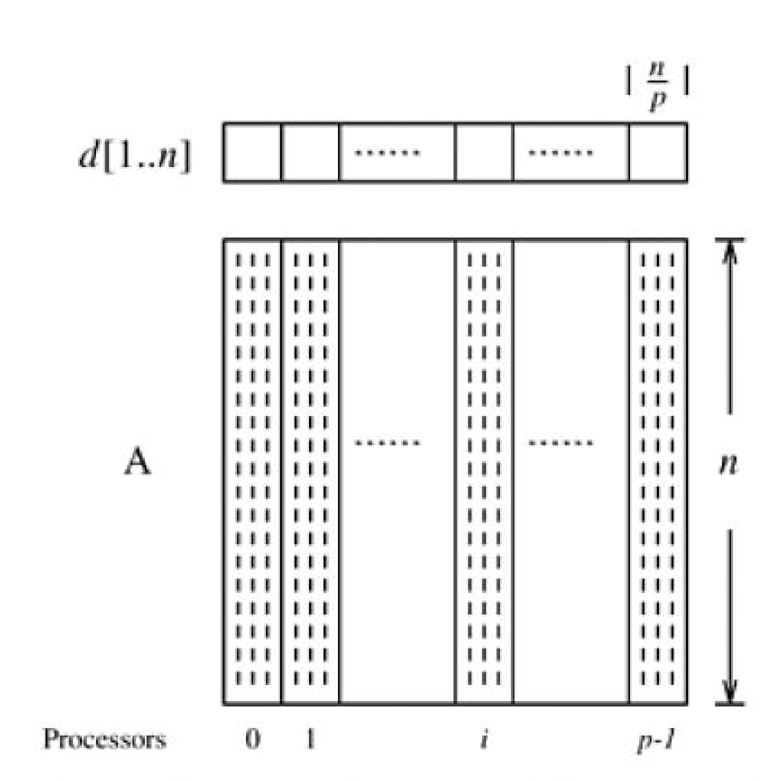
Минимальное остовное дерево
 - это остовное дерево этого графа, имеющее минимальный возможный вес.



Алгоритм Прима

```
procedure PRIM_MST(V, E, w, r)
begin
   V_T := {r};
   d[r] := 0;
   for all v ∈ (V - V_T ) do
      if edge (r, v) exists set d[v] := w(r, v);
      else set d[v] := ∞ ;
   while V_T ≠ V do
   begin
      find a vertex u such that d[u] :=min{d[v]|v ∈ (V - V_T )};
      V_T := V_T ∪ {u};
      for all v ∈ (V - V_T ) do
         d[v] := min{d[v], w(u, v)};
   endwhile
end PRIM_MST
```

Параллельный алгоритм Прима



Параллельный алгоритм Прима

На каждой итерации (пока множество V_t *!=* V):

1. На каждом P_i ищем ***min d*** до V_t
2. Редукция по ***min*** на 0 процессоре
3. Рассылка всем номера новой вершины включенной в V_t
4. Обновление величин ***d***

Параллельный алгоритм Прима

$$T_P = \overbrace{\Theta\left(\frac{n^2}{p}\right)}^{\text{computation}} + \overbrace{\Theta(n \log p)}^{\text{communication}}.$$

$$S = \frac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \log p)}$$

$$E = \frac{1}{1 + \Theta((p \log p)/n)}$$

Алгоритм Дейкстры

```
procedure DIJKSTRA_SINGLE_SOURCE_SP(V, E, w, s)
begin
   V_T := {s};
   for all v ∈ (V - V_T ) do
      if (s, v) exists set l[v] := w(s, v);
      else set l[v] := ∞ ;
   while V_T ≠ V do
   begin
      find a vertex u such that l[u] := min{l[v]|v ∈ (V - V_T )
      V_T := V_T ∪ {u};
      for all v ∈ (V - V_T ) do
         l[v] := min{l[v], l[u] + w(u, v)};
   endwhile
end DIJKSTRA_SINGLE_SOURCE_SP
```

Транзитивное замыкание

- $G = (V, E)$
- $G^* = (V, E^*)$, $E^* = \{(v_i, v_j) |$ путь от v_i до v_j в $G\}$
- $A^* = (a^*_{i,j})$
- $$a^*_{i,j} = \begin{cases} \infty & \text{if } d_{i,j} = \infty \\ 1 & \text{if } d_{i,j} > 0 \text{ or } i = j \end{cases}$$

Алгоритм Флойда - Уоршалла

```
procedure FLOYD_ALL_PAIRS_SP( A)
begin
   D(0) = A;
   for k := 1 to n do
       for i := 1 to n do
           for j := 1 to n do
```

$$d_{i,j}^{(k)} := \min\left(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right);$$

```
;
end FLOYD_ALL_PAIRS_SP
```

Параллельный алгоритм Флойда-Уоршалла

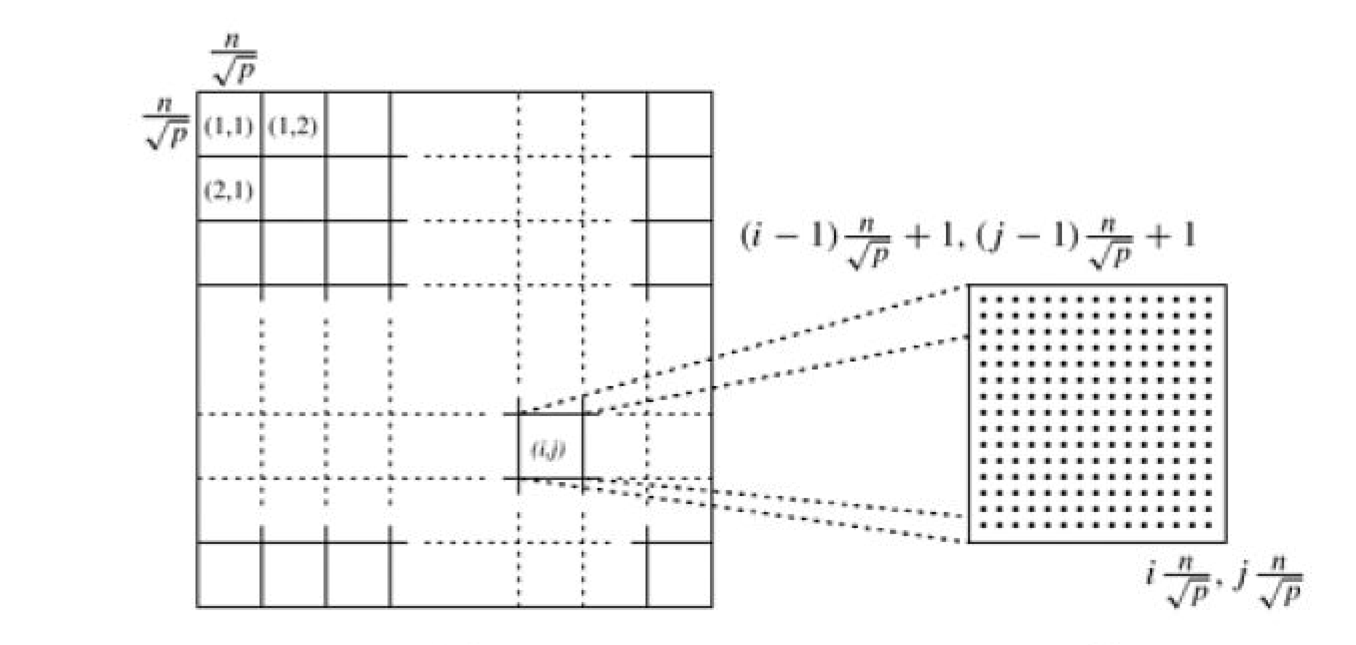
- Рассмотрим k-ю итерацию:

$$d^k_{i,j} = min\ (d_{i,j}, d_{i,k} + d_{k,j})$$

- На k-ой итерации $d_{i,k}$ *и* $d_{k,j}$ *меняться не будут:*

$$d^k_{k,j} = min\ (d_{k,j}, d_{k,k} + d_{k,j})$$
$$d^k_{i,k} = min\ (d_{i,k}, d_{i,k} + d_{k,k})$$

Параллельный алгоритм Флойда

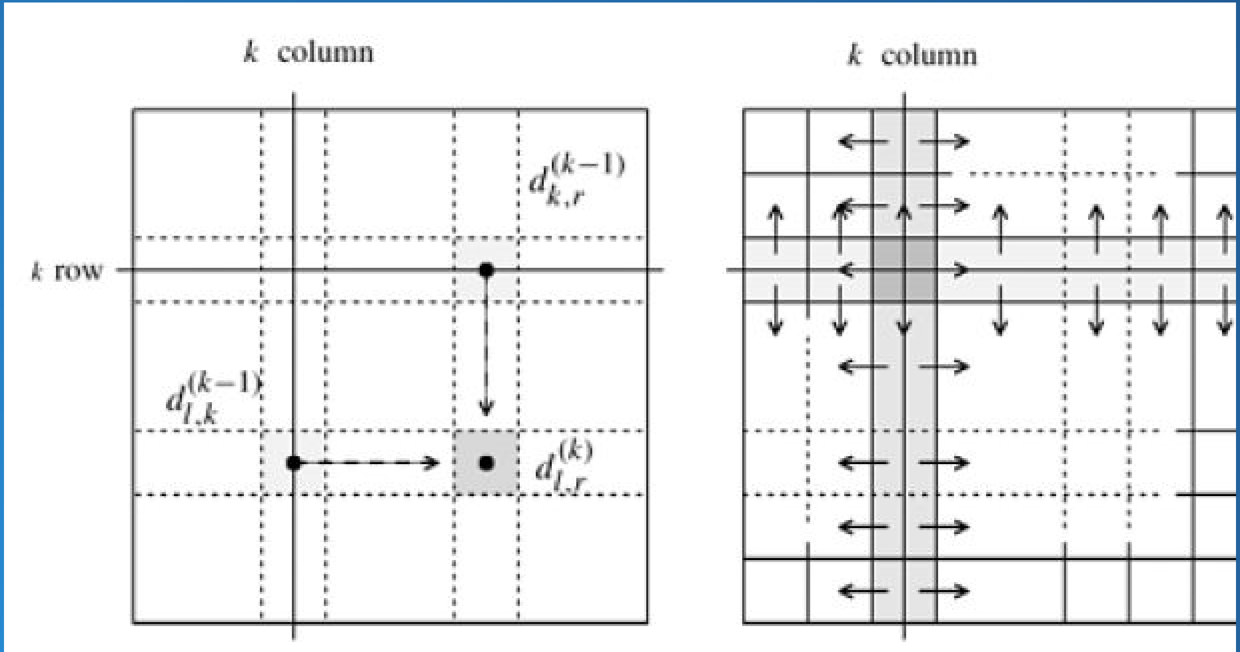


Параллельный алгоритм Флойда-Уоршалла

На каждой из n итераций по k

1. Каждый процесс $P_{i,j}$ содержащей k-ю строку рассылает ее $P_{*,j}$
2. Каждый процесс $P_{i,j}$ содержащей k-й столбец рассылает ее $P_{i,*}$
3. Локальный расчет свой части матрицы D на каждом процессе

Параллельный алгоритм Флойда-Уоршалла



Параллельный алгоритм Флойда-Уоршалла

$$T_P = \overbrace{\Theta\left(\frac{n^3}{p}\right)}^{\text{computation}} + \overbrace{\Theta\left(\frac{n^2}{\sqrt{p}}\log p\right)}^{\text{communication}}.$$

$$\begin{aligned} S &= \frac{\Theta(n^3)}{\Theta(n^3/p) + \Theta((n^2 \log p)/\sqrt{p})} \\ E &= \frac{1}{1 + \Theta((\sqrt{p} \log p)/n)} \end{aligned}$$