Сети и потоки. Алгоритм Диница

Кононов Николай

Математико-Механический факультет СПбГУ

2019

План

1. Сети и потоки
   - Простейшие понятия
   - Остаточная сеть, блокирующий поток
   - Теорема Форда-Фалкерсона
   - Слоистая сеть
2. Алгоритм Диница
   - Основные идеи
   - Пример
   - Корректность
   - Асимптотика и оценка числа фаз
   - Поиск блокирующего потока
   - Реализация

План

1. Сети и потоки
   - Простейшие понятия
   - Остаточная сеть, блокирующий поток
   - Теорема Форда-Фалкерсона
   - Слоистая сеть
2. Алгоритм Диница
   - Основные идеи
   - Пример
   - Корректность
   - Асимптотика и оценка числа фаз
   - Поиск блокирующего потока
   - Реализация

Сеть

- Определение: Пусть есть множество вершин V, в котором выделены две вершины: s (вход или исток) и t (выход или сток).
  Пусть определена функция $c : V \times V \to \mathbb{R}$, удовлетворяющая соотношениям

  $$\forall x, y \in V \quad c(x, y) \geq 0, \quad c(x, s) = 0, \quad c(t, y) = 0$$

  функция c - пропускная способность.

Сеть

- Определение: Пусть есть множество вершин V, в котором выделены две вершины: s (вход или исток) и t (выход или сток).
  Пусть определена функция $c : V \times V \to \mathbb{R}$, удовлетворяющая соотношениям

  $$\forall x, y \in V \quad c(x, y) \geq 0, \quad c(x, s) = 0, \quad c(t, y) = 0$$

  функция c - пропускная способность.
- $A = \{(x, y) : c(x, y) > 0\}$ - множество стрелок
  Тогда $G = ((V, A), s, t, c)$ - сеть

Поток в сети

- Определение: Пусть G– сеть, а функция $f : V \times V \to \mathbb{R}$ удовлетворяет трем условиям:
  1) $\forall x, y \in V \quad f(x, y) \leq c(x, y)$
  2) $\forall x, y \in V \quad f(x, y) = -f(y, x)$
  3) $\forall v \in V \backslash \{s, t\}$ выполняется условие: $\sum\limits_{x \in V} f(v, x) = 0$ - закон сохранения потока
  f - поток в сети G

Поток в сети

- Определение: Пусть $G-$ сеть, а функция $f : V \times V \to \mathbb{R}$ удовлетворяет трем условиям:
  1) $\forall x, y \in V \quad f(x, y) \le c(x, y)$
  2) $\forall x, y \in V \quad f(x, y) = -f(y, x)$
  3) $\forall v \in V \backslash \{s, t\}$ выполняется условие: $\sum\limits_{x \in V} f(v, x) = 0$ - закон сохранения потока
  f - поток в сети G
- $|f| = \sum\limits_{v \in V} f(s, v)$ - величина потока
  Поток с максимальной величиной - максимальный

Разрез сети

- Определение: пусть G - сеть, а множество ее вершин V разбито на два дизъюнктных множества $S \ni s$ и $T \ni t$. Тогда (S, T) - разрез сети G

Разрез сети

- Определение: пусть G - сеть, а множество ее вершин V разбито на два дизъюнктых множества $S \ni s$ и $T \ni t$. Тогда (S, T) - разрез сети G
- Величина $c(S, T) = \sum\limits_{x \in S, y \in T} c(x, y)$ называется пропускной способностью разреза.
  Любой разрез сети G с минимальной пропускной способностью называется минимальным.

Разрез сети

- Определение: пусть G - сеть, а множество ее вершин V разбито на два дизъюнктых множества $S \ni s$ и $T \ni t$. Тогда (S, T) - разрез сети G
- Величина $c(S, T) = \sum\limits_{x \in S, y \in T} c(x, y)$ называется пропускной способностью разреза.
  Любой разрез сети G с минимальной пропускной способностью называется минимальным.
- Для любого потока f величина $f(S, T) = \sum\limits_{x \in S, y \in T} f(x, y)$ называется потоком через разрез.

Разрез сети

- Определение: пусть G - сеть, а множество ее вершин V разбито на два дизъюнктых множества $S \ni s$ и $T \ni t$. Тогда (S, T) - разрез сети G
- Величина $c(S, T) = \sum\limits_{x \in S, y \in T} c(x, y)$ называется пропускной способностью разреза.
  Любой разрез сети G с минимальной пропускной способностью называется минимальным.
- Для любого потока f величина $f(S, T) = \sum\limits_{x \in S, y \in T} f(x, y)$ называется потоком через разрез.

Лемма

Лемма: Для любого потока f, и разреза (S, T) сети G выполняется $|f| = f(S, T)$

План

1. Сети и потоки
   - Простейшие понятия
   - Остаточная сеть, блокирующий поток
   - Теорема Форда-Фалкерсона
   - Слоистая сеть
2. Алгоритм Диница
   - Основные идеи
   - Пример
   - Корректность
   - Асимптотика и оценка числа фаз
   - Поиск блокирующего потока
   - Реализация

Остаточная сеть

- Остаточной пропускной способностью c_f по отношению к сети $G = \{(V, E), s, t, c\}$ и потоку f в ней называется пропускная способность

$$c_f(x, y) = \begin{cases} 0 & \text{if y = s or x = t} \\ c(x, y) - f(x, y) & \text{otherwise} \end{cases}$$

Остаточная сеть

- Остаточной пропускной способностью c_f по отношению к сети $G = \{(V, E), s, t, c\}$ и потоку f в ней называется пропускная способность

$$c_f(x, y) = \begin{cases} 0 & \text{if y = s or x = t} \\ c(x, y) - f(x, y) & \text{otherwise} \end{cases}$$

- Остаточной сетью для сети G и потока f называется сеть $G_f = \{(V, E_f), s, t, c_f\}$, где $E_f = \{(u, v) \in V \times V | c_f(u, v) > 0\}$
- Остаточное ребро можно интуитивно понимать как меру того, насколько можно еще увеличить поток вдоль этого ребра

Остаточная сеть

- Остаточной пропускной способностью c_f по отношению к сети $G = \{(V, E), s, t, c\}$ и потоку f в ней называется пропускная способность

$$c_f(x, y) = \begin{cases} 0 & \text{if y = s or x = t} \\ c(x, y) - f(x, y) & \text{otherwise} \end{cases}$$

- Остаточной сетью для сети G и потока f называется сеть $G_f = \{(V, E_f), s, t, c_f\}$, где $E_f = \{(u, v) \in V \times V | c_f(u, v) > 0\}$
- Остаточное ребро можно интуитивно понимать как меру того, насколько можно еще увеличить поток вдоль этого ребра

Определение

Простой st-путь в G_f называется дополняющим путем

Блокирующий поток

Определение

Блокирующим потоком f в сети $G = ((V, E), s, t, c)$ называется такой поток, что $\forall st$-путь содержит насыщенное этим потоком ребро. То есть в данной сети не найдется такого пути из истока в сток, вдоль которого можно безпрепятственно увеличить поток

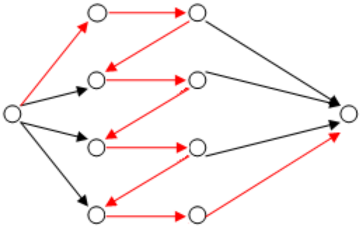
Замечание: блокирующий поток не всегда максимальный, более того, он может быть сколь угодно малым, относительно максимального

Блокирующий поток

Определение

Блокирующим потоком f в сети $G = ((V, E), s, t, c)$ называется такой поток, что $\forall st$-путь содержит насыщенное этим потоком ребро. То есть в данной сети не найдется такого пути из истока в сток, вдоль которого можно безпрепятственно увеличить поток

Замечание: блокирующий поток не всегда максимальный, более того, он может быть сколь угодно малым, относительно максимального
Пример: пропускная способность ребер 1, 'единичный' поток идет по красным ребрам



План

Теорема Форда-Фалкерсона

Theorem

Ford-Fulkerson: В сети G с пропускной способностью c задан поток f, тогда следующие три утверждения равносильны:
1) Поток f максимален
2) $\exists (S, T) : |f| = c(S, T)$
3) В остаточной сети G_f нет дополняющего пути

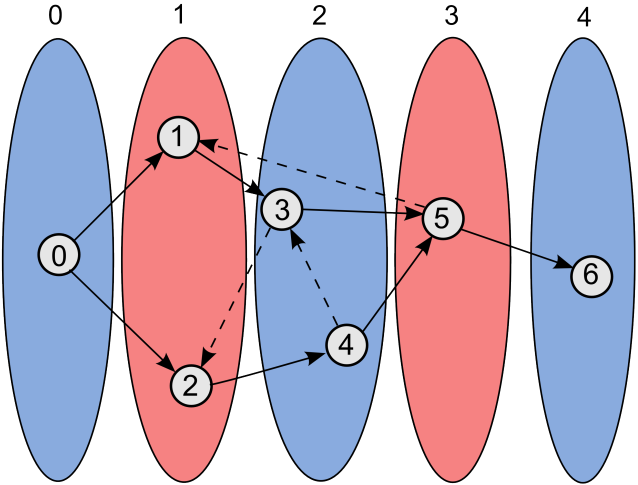
План

Слоистая сеть

Слоистая сеть(layered network, вспомогательная сеть) строится след образом:

- Для каждой вершины v данной сети G определим длину кратчайшего sv-пути из истока и обозначим ее d[v] (можно сделать обходом в ширину)
- В слоистую сеть включаем только стрелки (u, v) такие, что d[u] = d[v] + 1
- То есть исключим из G стрелки лежащие внутри одного уровня или идущие назад
- Получившаяся сеть ациклична и любой $s \rightsquigarrow t$ путь в слоистой сети является кратчайшим путем в исходной сети из свойств BFS
- $G = \{\{1, 2\}, \{3\}, \{4\}, \{2, 5\}, \{3, 5\}, \{1, 6\}\}; \quad s = 0, t = 6$ тогда слоистая сеть $G_s = \{\{1, 2\}, \{3\}, \{4\}, \{5\}, \{5\}, \{6\}\}$

Пример слоистой сети



План

Алгоритм Диница. Основные идеи

Постановка задачи

Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа

Алгоритм Диница. Основные идеи

Постановка задачи

Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа
- Изначально пусть $f(e) = 0 \quad \forall e \in E$

Алгоритм Диница. Основные идеи

Постановка задачи

Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа
- Изначально пусть $f(e) = 0 \quad \forall e \in E$
- Алгоритм состоит из нескольких фаз.

Алгоритм Диница. Основные идеи

Постановка задачи

Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа
- Изначально пусть $f(e) = 0 \quad \forall e \in E$
- Алгоритм состоит из нескольких фаз.
- На каждой фазе строится остаточная сеть G_f, затем по отношению к G_f строится слоистая сеть G_L(BFS). Если d[t] $= \infty$ останавливаемся и выводим f

Алгоритм Диница. Основные идеи

Постановка задачи

Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа
- Изначально пусть $f(e) = 0 \quad \forall e \in E$
- Алгоритм состоит из нескольких фаз.
- На каждой фазе строится остаточная сеть G_f, затем по отношению к G_f строится слоистая сеть G_L(BFS). Если d[t] $=\infty$ останавливаемся и выводим f
- В построенной слоистой сети находим блокирующий поток $f‘$ (любой)

Алгоритм Диница. Основные идеи

Постановка задачи

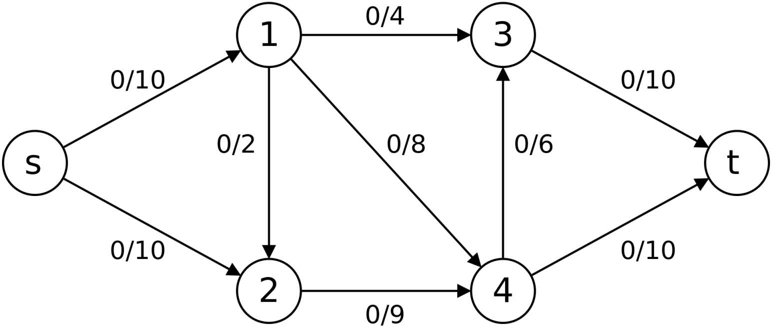
Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- Алгоритм является улучшенной версией Алгоритма Эдмонса-Карпа
- Изначально пусть $f(e) = 0 \quad \forall e \in E$
- Алгоритм состоит из нескольких фаз.
- На каждой фазе строится остаточная сеть G_f, затем по отношению к G_f строится слоистая сеть G_L(BFS). Если d[t] $=\infty$ останавливаемся и выводим f
- В построенной слоистой сети находим блокирующий поток f‘ (любой)
- Дополняем поток f потоком f‘ и переходим к следующей фазе

План

Пример

- $f = 0$
- $(G, f) \to G_f \to G_L \to f' \to f = f + f$‘



План

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Доказательство.

Предположим, что в какой-то момент в слоистой сети G_L построенной для остаточной сети G_f не удалось найти блокирующий поток.

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Доказательство.

Предположим, что в какой-то момент в слоистой сети G_L построенной для остаточной сети G_f не удалось найти блокирующий поток.
Это означает, что $\mathrm{d}[\mathrm{t}] = \infty$, то есть сток t не достижим из истока s в слоистой сети .

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Доказательство.

Предположим, что в какой-то момент в слоистой сети G_L построенной для остаточной сети G_f не удалось найти блокирующий поток.
Это означает, что $\mathrm{d[t]} = \infty$, то есть сток t не достижим из истока s в слоистой сети .
Но слоистая сеть содержит в себе все кратчайшие пути в сети G_f из истока s.

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Доказательство.

Предположим, что в какой-то момент в слоистой сети G_L построенной для остаточной сети G_f не удалось найти блокирующий поток.
Это означает, что $\mathrm{d}[\mathrm{t}] = \infty$, то есть сток t не достижим из истока s в слоистой сети .
Но слоистая сеть содержит в себе все кратчайшие пути в сети G_f из истока s.
Таким образом в остаточной сети нет $s \rightsquigarrow t$ пути

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины.

Доказательство.

Предположим, что в какой-то момент в слоистой сети G_L построенной для остаточной сети G_f не удалось найти блокирующий поток.
Это означает, что $\mathrm{d[t]} = \infty$, то есть сток t не достижим из истока s в слоистой сети .
Но слоистая сеть содержит в себе все кратчайшие пути в сети G_f из истока s.
Таким образом в остаточной сети нет $s \rightsquigarrow t$ пути
Применяя теорему Форда-Фалкерсона получаем, что текущий поток в самом деле максимален. □

План

Lemma

Кратчайшее расстояние между истоком и стоком устрого увеличивается с выполнением каждой итерации: $d_i[t] > d_{i-1}[t] \quad \forall i$

Доказательство.

От противного. Пусть длина кратчайшего $s \rightsquigarrow t$ пути не изменилась после i-ой итерации. Слоистая сеть G_L строится по остаточной G_f. Рассмотрим кратчайший $s \rightsquigarrow t$ путь. По предположению его длина должна остаться неизменной. Однако G_f^i содержит только ребра остаточной сети перед i-й фазой, либо обратные к ним.

Lemma

Кратчайшее расстояние между истоком и стоком устрого увеличивается с выполнением каждой итерации: $d_i[t] > d_{i-1}[t] \quad \forall i$

Доказательство.

От противного. Пусть длина кратчайшего $s \rightsquigarrow t$ пути не изменилась после i-ой итерации. Слоистая сеть G_L строится по остаточной G_f.
Рассмотрим кратчайший $s \rightsquigarrow t$ путь. По предположению его длина должна остаться неизменной. Однако G_f^i содержит только ребра остаточной сети перед i-й фазой, либо обратные к ним.
Таким образом пришли к противоречию: нашелся $s \rightsquigarrow t$ путь, который не содержит насыщенных ребер, и имеет ту же длину, что и кратчайший путь. Этот путь должен был быть "заблокирован"блокирующим потоком. □

Lemma

Кратчайшее расстояние от истока до каждой вершины не уменьшается с выполнением каждой итерации:
$\forall v \in V \quad d_i[v] \geq d_{i-1}[v]$

Доказательство.

Расмотрим произвольные v и i и кратчайший $s \rightsquigarrow v$-путь P в сети G_f^i. $|P| = d_i[v]$ Заметим, что в остаточную сеть G_f^i могут входить стрелки G_f, а также стрелки обратные к ним.

Lemma

Кратчайшее расстояние от истока до каждой вершины не уменьшается с выполнением каждой итерации:
$\forall v \in V \quad d_i[v] \geq d_{i-1}[v]$

Доказательство.

Расмотрим произвольные v и i и кратчайший $s \rightsquigarrow v$-путь P в сети G_f^i. $|P| = d_i[v]$ Заметим, что в остаточную сеть G_f^i могут входить стрелки G_f, а также стрелки обратные к ним. Рассмотрим 2 случая:

Lemma

Кратчайшее расстояние от истока до каждой вершины не уменьшается с выполнением каждой итерации:
$\forall v \in V \quad d_i[v] \geq d_{i-1}[v]$

Доказательство.

Расмотрим произвольные v и i и кратчайший $s \rightsquigarrow v$-путь P в сети G_f^i. $|P| = d_i[v]$ Заметим, что в остаточную сеть G_f^i могут входить стрелки G_f, а также стрелки обратные к ним. Рассмотрим 2 случая:

- Путь P содержит только ребра из G_f. Тогда $|P| \geq d_i[v]$ ($d_i[v]$ - длина кратчайшего пути) $\Longleftrightarrow d_i[v] \geq d_{i-1}[v]$

□

Окончание доказательства

Доказательство.

- Путь P содержит хотя бы одно ребро, не содержащееся в сети G_f, но обратное какому-то из ее ребер. Рассмотрим первое такое ребро (u, w) в пути P: $s \Longrightarrow u \rightarrow v \Longrightarrow t$ Применим лемму к вершине u, т.к. она удовлетворяет условию первого случая: $d_i[u] \geq d_{i-1}[u](1)$

Окончание доказательства

Доказательство.

- Путь P содержит хотя бы одно ребро, не содержащееся в сети G_f, но обратное какому-то из ее ребер. Рассмотрим первое такое ребро (u, w) в пути P: $s \Longrightarrow u \rightarrow v \Longrightarrow t$ Применим лемму к вершине u, т.к. она удовлетворяет условию первого случая: $d_i[u] \geq d_{i-1}[u]$(1) Теперь заметим, что т.к. (u, w) появилось в остаточной сети только после выполнения (i-1)-ой фазы $\Rightarrow$ вдоль ребра (w, u) был дополнительно пропущен какой-то поток. Следовательно, ребро (w, u) пренадлежало слоистой сети перед (i-1)-й фазой $\Rightarrow$d$_{i-1}[u] = d_{i-1}[v] + 1$(2) По свойству кратчайших путей: $d_i[w] = d_i[u] + 1$(3) Объединяя (1), (2), (3) получим: $d_i[w] \geq d_{i-1}[w] + 2$. Теперь мы можем применять те же рассуждения ко всему оставшемуся пути до v и получить требуемое неравенство

□

Оценка числа фаз

- Так как длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n - 1 \Rightarrow$ алгоритм Диница совершает не больше $n - 1$ фазы (итераций цикла).

Оценка числа фаз

- Так как длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n - 1 \Rightarrow$ алгоритм Диница совершает не больше $n - 1$ фазы (итераций цикла).
- Таким образом, в зависимости от того, каким алгоритмом нахождения блокирующего потока мы пользовались алгоритм Диница может выполнятся за $O(|V| \cdot |E|^2)$ или за $O(|V|^2 \cdot |E|)$

Оценка числа фаз

- Так как длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n-1 \Rightarrow$ алгоритм Диница совершает не больше $n-1$ фазы (итераций цикла).
- Таким образом, в зависимости от того, каким алгоритмом нахождения блокирующего потока мы пользовались алгоритм Диница может выполнятся за $O(|V| \cdot |E|^2)$ или за $O(|V|^2 \cdot |E|)$
- Возможно достичь асимптотики $O(|V| \cdot |E| \cdot log(|V|)$, используя динамические деревья Слетора и Тарьяна

План

- Так как слоистая сеть G_L, в которой ищется блокирующий поток ациклическая - будем искать блокирующий поток в ациклической сети.
- Искать $s \rightsquigarrow t$ пути по одному, пока такие пути находятся
- DFS найдет все $s \rightsquigarrow t$ пути, если t достижима из s, а $c(u, v) > 0 \quad \forall (u, v) \in E$
- Насыщая ребра, мы хотя бы единожды достигнем стока t, следовательно блокирующий поток всегда найдется.
- DFS находит каждый путь за O(E), каждый путь насыщает как минимум одно ребро $\Rightarrow O(E)$
  Итоговая асимптотика: $\mathrm{O}(E^2)$

Оптимизация. Удаляющий обход

- Будем использовать предыдущий алгоритм, удаляя при этом ребра, из которых невозможно дойти до стока t

Оптимизация. Удаляющий обход

- Будем использовать предыдущий алгоритм, удаляя при этом ребра, из которых невозможно дойти до стока t
- Достаточно удалять ребро после того, как мы просмотрели его в DFS, если не нашелся путь до стока

Оптимизация. Удаляющий обход

- Будем использовать предыдущий алгоритм, удаляя при этом ребра, из которых невозможно дойти до стока t
- Достаточно удалять ребро после того, как мы просмотрели его в DFS, если не нашелся путь до стока
- Будем поддерживать в списке смежности каждой вершины указатель на первое удаленное ребро и увеличивать его внутри цикла DFS

Оптимизация. Удаляющий обход

- Будем использовать предыдущий алгоритм, удаляя при этом ребра, из которых невозможно дойти до стока t
- Достаточно удалять ребро после того, как мы просмотрели его в DFS, если не нашелся путь до стока
- Будем поддерживать в списке смежности каждой вершины указатель на первое удаленное ребро и увеличивать его внутри цикла DFS
- Если DFS достигает стока: насыщается как минимум одно ребро. Иначе как минимум один указатель продвигается вперед. Значит один запуск обхода в глубину работает за O(V + K), K - число продвижения указателей. Всего запусков DFS для поиска блокирующего потока: O(P), где P - количество ребер, насыщенных блокирующим потоком. Таким образом весь алгоритм отработает за $O(P \cdot V + \sum_i K_i = O(P \cdot V + E)$. В худшем случае, когда P = E, $O(V \cdot E)$

План

```
int dfs(int v, int flow)
  if (flow == 0)
    return 0
  if (v == t)
    return flow
  for (u = ptr[v] to n)
    if (vu ∈ E)
      pushed = dfs(u, min(flow, c(vu) - f(vu)))
      f(vu) += pushed
      f(uv) -= pushed
      return pushed
    ptr[v]++
  return 0
```

Реализация

```
main()
   ...
   flow = 0
   for (int i = 1 to n)
      ptr[i] = 0
   do
      pushed = dfs(S, ∞)
      flow += pushed
   while (pushed > 0)
```

Литература I

Т. Кормен.
Алгоритмы. Построение и анализ.
Глава 27, "Максимальный поток".

neerc.ifmo.ru
"Схема алгоритма Диница"