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

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

тогда 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

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

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

Поток в сети

- **Определение:** Пусть G - сеть, а функция $f : V \times V \to \mathbb{R}$ удовлетворяет трем аксиомам:
  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) $\forall v \in V \backslash \{s,t\} \quad \sum\limits_{x \in V} f(v,x) = 0$ - закон сохранения потока

  Тогда f - **поток в сети** G

Поток в сети

- **Определение:** Пусть G - сеть, а функция $f: V \times V \to \mathbb{R}$ удовлетворяет трем аксиомам:
  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) $\forall v \in V \backslash \{s, t\} \quad \sum\limits_{x \in V} f(v, x) = 0$ - закон сохранения потока

  Тогда 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 дизъюнктых 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 дизъюнктных 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

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

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

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

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

- **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

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

- **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 $s \rightsquigarrow t$ 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

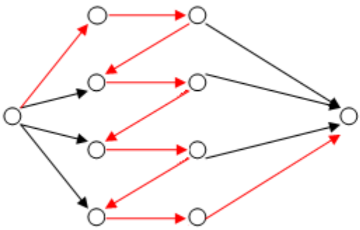
Замечание: блокирующий поток не всегда максимальный. Более того, он может быть сколь угодно малым, относительно максимального

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 $s \rightsquigarrow v$ пути и обозначим ее `d[v]` (можно сделать обходом в ширину)

Слоистая сеть

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

- Для каждой вершины v данной сети G определим длину кратчайшего $s \rightsquigarrow v$ пути 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`

Слоистая сеть

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

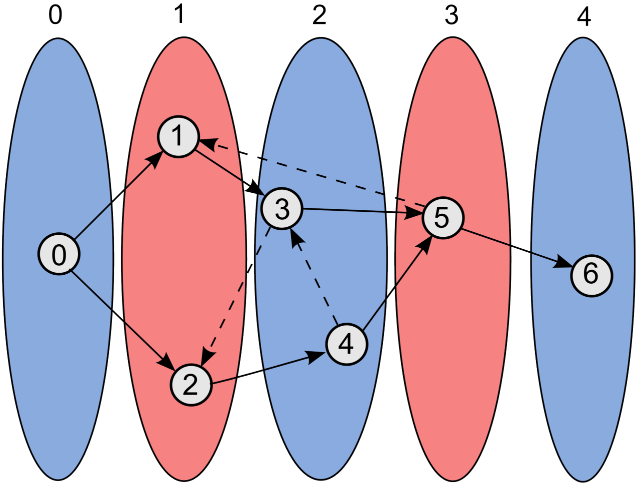
- Для каждой вершины v данной сети G определим длину кратчайшего $s \rightsquigarrow v$ пути и обозначим ее `d[v]` (можно сделать обходом в ширину)
- В слоистую сеть включаем только стрелки (u, v) такие, что `d[u] = d[v] + 1`
- То есть исключим из G стрелки лежащие внутри одного уровня или идущие назад

Слоистая сеть

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

- Для каждой вершины v данной сети G определим длину кратчайшего $s \rightsquigarrow v$ пути 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**

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 Диница. Основные идеи

Постановка задачи

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

- 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?

- 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?

- 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] = ` ∞ 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?

- 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] = ` ∞ останавливаемся и выводим f
- В построенной слоистой сети находим блокирующий поток f‘ (любой)

Алгоритм Диница. Основные идеи

Постановка задачи

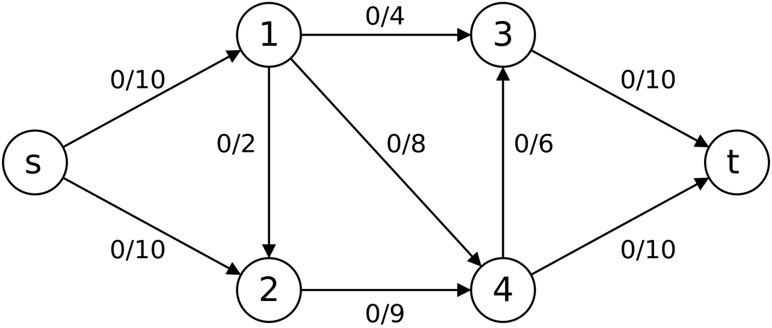
Пусть дана сеть $G = ((V, E), s, t, c)$. Как найти поток f из s в t максимальной величины?

- 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] = ` ∞ 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

- $\mathrm{f} = 0$
- $(G, f) \to G_f \to G_L \to f' \to f = f + f'$



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

Proof

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

Proof

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 `d[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

Proof

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 `d[t] =` ∞, 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

Proof

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 `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

Корректность алгоритма

Theorem

Если алгоритм завершается, полученный поток является потоком максимальной длины

Proof

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 `d[t] = ` ∞, 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. $\square$

План

Lemma

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

Proof

От противного. Пусть длина кратчайшего $s \rightsquigarrow t$ пути не изменилась после i-ой итерации. Слоистая сеть G_L строится по остаточной G_f.

Lemma

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

Proof

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 строго увеличивается с выполнением каждой итерации: $d_i[t] > d_{i-1}[t] \quad \forall i$

Proof

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. □

Оценка числа фаз

- Длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n - 1 \Rightarrow$ алгоритм Диница совершает не больше $n - 1$ итераций цикла.

Оценка числа фаз

- Длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n - 1 \Rightarrow$ алгоритм Диница совершает не больше $n - 1$ 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

- Длина кратчайшего $s \rightsquigarrow t$ пути не может превосходить $n - 1 \Rightarrow$ алгоритм Диница совершает не больше $n - 1$ 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.

- Так как слоистая сеть G_L, в которой ищется блокирующий поток ациклическая - будем искать блокирующий поток в ациклической сети.
- Будем искать $s \rightsquigarrow t$ пути по одному, пока такие пути находятся

- Так как слоистая сеть G_L, в которой ищется блокирующий поток ациклическая - будем искать блокирующий поток в ациклической сети.
- Будем искать $s \rightsquigarrow t$ пути по одному, пока такие пути находятся
- DFS найдет все $s \rightsquigarrow t$ пути, если t достижима из s, а $c(u, v) > 0 \quad \forall (u, v) \in E$

- Так как слоистая сеть G_L, в которой ищется блокирующий поток ациклическая - будем искать блокирующий поток в ациклической сети.
- Будем 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

- Так как слоистая сеть G_L, в которой ищется блокирующий поток ациклическая - будем искать блокирующий поток в ациклической сети.
- Будем 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: $O(E^2)$

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

Реализация

```
bool bfs():
  заполняем массив d значениями, равными ∞
  d[s] = 0
  Q.push(s)
  while !Q.isEmpty
    u = Q.pop()
    for (uv) ∈ E(G)
      if f[u][v] < c[u][v] and d[v] == ∞
        d[v] = d[u] + 1
        Q.push(v)
  return d[t] != ∞
```

Реализация

```
// поиск блокирующего потока
// u — номер вершины
// minC — минимальная пропускная способность дополняющей сети на пройденно
int dfs(u, minC):
  if u == t or minC == 0
    return minC
  for v = p[u] to |V(G)| − 1
    if d[v] == d[u] + 1 // это условие эквивалентно поиску во вспомогател
      delta = dfs(v, min(minC, c[u][v] - f[u][v]))
      if delta != 0
        f[u][v] += delta // насыщаем рёбра по пути dfs
        f[v][u] -= delta
        return delta
    p[u]++
  return 0
```

Реализация

```
int findMaxFlow():
  maxFlow = 0
  while bfs() // пересчитываем d[i], заодно проверяем достижима ли t из s
    заполняем p нулями
    flow = dfs(s, ∞)
    while flow != 0
      maxFlow += flow
      flow = dfs(s, ∞)
  return maxFlow
```

Литература I

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

neerc.ifmo.ru
"Схема алгоритма Диница"