Алгоритмы для NP-трудных задач

Лекция 11: FPT-алгоритмы

А. Куликов

Computer Science клуб при ПОМИ
http://logic.pdmi.ras.ru/∼infclub/

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

Лемма о подсолнухе

Определение

Подсолнухом с k лепестками (sunflower with k petals) называется семейство из k (конечных) множеств, пересечения любых двух их которых совпадают. Это пересечение называется центром (core) цветка:

$$S_i \cap S_j = C \text{ для всех } 1 \leq i < j \leq k\,.$$

Лемма о подсолнухе

Определение

Подсолнухом с k лепестками (sunflower with k petals) называется семейство из k (конечных) множеств, пересечения любых двух их которых совпадают. Это пересечение называется центром (core) цветка:

$$S_i \cap S_j = C \text{ для всех } 1 \leq i < j \leq k\,.$$

Лемма о подсолнухе (Sunflower Lemma)

В любом семействе, состоящем из более чем $s!(k-1)^s$ множеств размера не более s, найдется подсолнух с хотя бы k лепестками.

Доказательство

Доказательство

- Индукция по s.

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).
- Пусть теперь $s \geq 2$ и $\mathcal{F}$ — семейство, содержащее более $s!(k-1)^s$ множеств размера не более s.

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).
- Пусть теперь $s \geq 2$ и $\mathcal{F}$ — семейство, содержащее более $s!(k-1)^s$ множеств размера не более s.
- Рассмотрим максимальное семейство попарно не пересекающихся множеств $\mathcal{A} = \{A_1, \ldots, A_t\}$.

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).
- Пусть теперь $s \geq 2$ и $\mathcal{F}$ — семейство, содержащее более $s!(k-1)^s$ множеств размера не более s.
- Рассмотрим максимальное семейство попарно не пересекающихся множеств $\mathcal{A} = \{A_1, \ldots, A_t\}$.
- Если $t \geq k$, то необходимый подсолнух найден.

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).
- Пусть теперь $s \geq 2$ и $\mathcal{F}$ — семейство, содержащее более $s!(k-1)^s$ множеств размера не более s.
- Рассмотрим максимальное семейство попарно не пересекающихся множеств $\mathcal{A} = \{A_1, \dots, A_t\}$.
- Если $t \geq k$, то необходимый подсолнух найден.
- Пусть $t \leq k - 1$ и $B = A_1 \cup \dots \cup A_t$. Тогда $|B| \leq s(k-1)$ и B пересекает любое множество семейства $\mathcal{F}$.

Доказательство

- Индукция по s.
- При $s = 1$ есть более $k - 1$ одноэлементных множеств. Любые k из них составляют подсолнух (с пустым центром).
- Пусть теперь $s \geq 2$ и $\mathcal{F}$ — семейство, содержащее более $s!(k-1)^s$ множеств размера не более s.
- Рассмотрим максимальное семейство попарно не пересекающихся множеств $\mathcal{A} = \{A_1, \ldots, A_t\}$.
- Если $t \geq k$, то необходимый подсолнух найден.
- Пусть $t \leq k - 1$ и $B = A_1 \cup \cdots \cup A_t$. Тогда $|B| \leq s(k-1)$ и B пересекает любое множество семейства $\mathcal{F}$.
- Значит, найдется элемент $x \in B$, содержащийся в хотя бы

$$\frac{|\mathcal{F}|}{|B|} > \frac{s!(k-1)^s}{s(k-1)} = (s-1)!(k-1)^{s-1}$$

множествах семейства $\mathcal{F}$.

Доказательство (продолжение)

Доказательство (продолжение)

- По индукционному предположению семейство

$$\mathcal{F}_x = \{S \setminus \{x\} \mid S \in \mathcal{F}, x \in S\}$$

содержит подсолнух с k лепестками.

Доказательство (продолжение)

- По индукционному предположению семейство

$$\mathcal{F}_x = \{S \setminus \{x\} \mid S \in \mathcal{F}, x \in S\}$$

  содержит подсолнух с k лепестками.
- Добавив x к множествам этого подсолнуха, получим подсолнух исходного семейства. □

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

Задача о d-множестве представителей

Задача о d-множестве представителей

- В задаче о d-множестве представителей (d-hitting set problem) по данному семейству множеств размера не более d каждое необходимо найти множество из не более k элементов, пересекающее каждое множество семейства.

Задача о d-множестве представителей

- В задаче о d-множестве представителей (d-hitting set problem) по данному семейству множеств размера не более d каждое необходимо найти множество из не более k элементов, пересекающее каждое множество семейства.
- Правило упрощения: если $k+1$ множество образуют подсолнух, заменить их всех на центр цвета. Если подсолнуха нет, то семейство содержит не более $O(k^d)$ множеств.

Задача о d-множестве представителей

- В задаче о d-множестве представителей (d-hitting set problem) по данному семейству множеств размера не более d каждое необходимо найти множество из не более k элементов, пересекающее каждое множество семейства.
- Правило упрощения: если $k+1$ множество образуют подсолнух, заменить их всех на центр цвета. Если подсолнуха нет, то семейство содержит не более $O(k^d)$ множеств.
- Получили ядро для задачи о минимальном множестве представителей, а значит, она принадлежит классу FPT.

План лекции

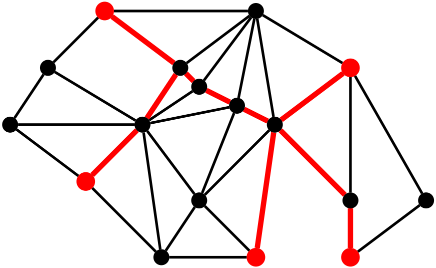
1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

План лекции

Steiner tree problem

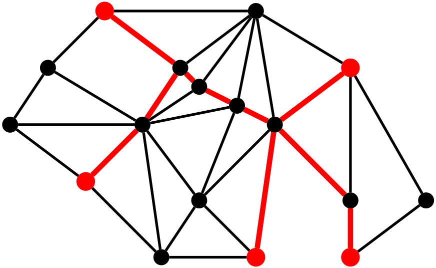
Steiner tree problem

- Steiner tree problem заключается в нахождении по данному графу G и множеству S из k его вершин дерева T минимального веса, содержащего все вершины множества S. Вершины множества S называются терминалами (termianl).



Steiner tree problem

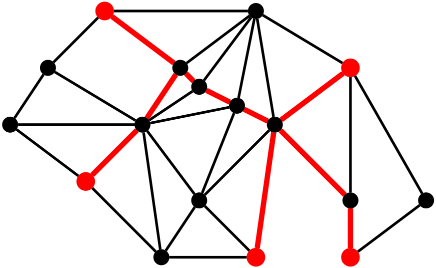
- Steiner tree problem заключается в нахождении по данному графу G и множеству S из k его вершин дерева T минимального веса, содержащего все вершины множества S. Вершины множества S называются терминалами (termianl).



- Задача очень похожа на задачу от минимальном покрывающем дереве, но, в отличие от последней, NP-трудна.

Steiner tree problem

- Steiner tree problem заключается в нахождении по данному графу G и множеству S из k его вершин дерева T минимального веса, содержащего все вершины множества S. Вершины множества S называются терминалами (termianl).



- Задача очень похожа на задачу от минимальном покрывающем дереве, но, в отличие от последней, NP-трудна.
- Мы покажем, что при параметризации $k = |S|$ она принадлежит классу FPT.

Решение методом динамического программирования

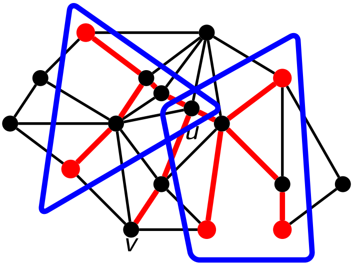
Лемма

Для $v \in V$ и $X \subseteq S$ пусть $c(v, X)$ — минимальная стоимость дерева для X, содержащего v, $d(u, v)$ — расстояние от u до v. Тогда

$$c(v, X) = \min_{u \in V, \emptyset \subset X' \subset X} c(u, X' \setminus u) + c(u, X \setminus X' \setminus u) + d(u, v).$$

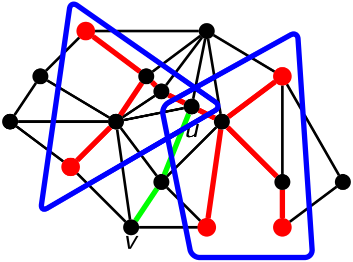
Доказательство леммы

$$c(v,X)=\min_{u\in V,\emptyset\subset X'\subset X} c(u,X'\setminus u)+c(u,X\setminus X'\setminus u)+d(u,v).$$



$\leq$: Дерево T_1, на котором достигается $c(u,X'\setminus u)$, дерево T_2, на котором достигается $c(u,X\setminus X'\setminus u)$, и путь из u в v дают в объединении дерево, содержащее все вершины множества X и v.

Доказательство леммы



$\leq$: Пусть на дереве T достигается $c(v, X)$ и пусть T' — его минимальное дерево, покрывающее X. Пусть u — вершина T', ближайшая к v. Найдется компонента в $T \setminus u$, содержащая подмножество терминалов $\emptyset \subset X' \subset X$. Значит, T есть объединение дерева, содержащего $X' \setminus u$ и u, дерева, содержащего $X \setminus X' \setminus u$ и u, и пути из u в v.

Алгоритм

Алгоритм

- Итак, достаточно заполнить матрицу размера $2^k|V|$ в порядке возрастания размера $|X|$.

Алгоритм

- Итак, достаточно заполнить матрицу размера $2^k|V|$ в порядке возрастания размера $|X|$.
- Для вычисления значения $c(v, X)$ необходимо перебрать $2^{|X|}$ значений.

Алгоритм

- Итак, достаточно заполнить матрицу размера $2^k|V|$ в порядке возрастания размера $|X|$.
- Для вычисления значения $c(v, X)$ необходимо перебрать $2^{|X|}$ значений.
- Время работы:

$$\sum_{X \subseteq S} 2^{|X|} = \sum_{i=1}^{k} \binom{k}{i} 2^i \leq (1+2)^k = 3^k$$

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

План лекции

1. Лемма о подсолнухе
   - Задача о минимальном множестве представителей
2. Динамическое программирование
   - Steiner tree problem
3. Итеративное сжатие
   - Задача об удалении нечетных циклов

Задача об удалении нечетных циклов

Задача об удалении нечетных циклов

- Задача об удалении нечетных циклов (odd cycle transversal problem, bipartite deletion problem, graph bipartization problem) заключается в проверке по данному графу G и числу k существования в графе таких k вершин, после удаления которых граф становится двудольным. Такое множество вершин S называется секущим (odd cycle transversal).

Задача об удалении нечетных циклов

- Задача об удалении нечетных циклов (odd cycle transversal problem, bipartite deletion problem, graph bipartization problem) заключается в проверке по данному графу G и числу k существования в графе таких k вершин, после удаления которых граф становится двудольным. Такое множество вершин S называется секущим (odd cycle transversal).
- Compression version задачи об удалении нечетных циклов: дан граф G и его секущее множество S размера $k+1$ и необходимо проверить, есть ли у G секущее множество размера k.

Итеративное сжатие

Итеративное сжатие

- Построив алгоритм Compression для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.

Итеративное сжатие

- Построив алгоритм $\textsc{Compression}$ для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.
- Пусть $V = \{v_1, \ldots, v_n\}$ и $G_i = G[V_i]$.

Итеративное сжатие

- Построив алгоритм Compression для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.
- Пусть $V = \{v_1, \ldots, v_n\}$ и $G_i = G[V_i]$.
- Простые замечания:

Итеративное сжатие

- Построив алгоритм $\textsc{Compression}$ для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.
- Пусть $V = \{v_1, \ldots, v_n\}$ и $G_i = G[V_i]$.
- Простые замечания:
  - если S является секущим для G, то оно является секущим для всех G_i;

Итеративное сжатие

- Построив алгоритм Compression для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.
- Пусть $V = \{v_1, \ldots, v_n\}$ и $G_i = G[V_i]$.
- Простые замечания:
  - если S является секущим для G, то оно является секущим для всех G_i;
  - если S является секущим для G_i, то $S \cup \{v_{i+1}\}$ является секущим для G_{i+1};

Итеративное сжатие

- Построив алгоритм Compression для compression version задачи об удалении нечетных циклов со временем работы $f(k)n^c$, мы сможем решить исходную задачу за $f(k)n^{c+1}$.
- Пусть $V = \{v_1, \dots, v_n\}$ и $G_i = G[V_i]$.
- Простые замечания:
  - если S является секущим для G, то оно является секущим для всех G_i;
  - если S является секущим для G_i, то $S \cup \{v_{i+1}\}$ является секущим для G_{i+1};
  - V_k — секущее для G_k.

Итеративное сжатие

Итеративное сжатие

- Запустим $\textsc{Compression}(G_{k+1}, V_{k+1}, k)$.

Итеративное сжатие

- Запустим $\textsc{Compression}(G_{k+1}, V_{k+1}, k)$.
- Если ответом будет "нет", то секущего множества размера k нет и для исходного графа.

Итеративное сжатие

- Запустим $\textsc{Compression}(G_{k+1}, V_{k+1}, k)$.
- Если ответом будет "нет", то секущего множества размера k нет и для исходного графа.
- Если же ответом будет секущее множество S_{k+1} размера k, то запустим $\textsc{Compression}(G_{k+2}, S_{k+1} \cup \{v_{k+2}\}, k)$.

Итеративное сжатие

- Запустим $\textsc{Compression}(G_{k+1}, V_{k+1}, k)$.
- Если ответом будет "нет", то секущего множества размера k нет и для исходного графа.
- Если же ответом будет секущее множество S_{k+1} размера k, то запустим $\textsc{Compression}(G_{k+2}, S_{k+1} \cup \{v_{k+2}\}, k)$.
- И так далее до G_n.

Итеративное сжатие

- Запустим $\textsc{Compression}(G_{k+1}, V_{k+1}, k)$.
- Если ответом будет “нет”, то секущего множества размера k нет и для исходного графа.
- Если же ответом будет секущее множество S_{k+1} размера k, то запустим $\textsc{Compression}(G_{k+2}, S_{k+1} \cup \{v_{k+2}\}, k)$.
- И так далее до G_n.
- Мы предъявим алгоритм, решающий compression version за время $O(3^k k|E|)$.

Решение compression version

Решение compression version

- Итак, пусть дан граф G и его секущее множество S' размера $k+1$ и необходимо проверить, есть ли секущее множество S размера k.

Решение compression version

- Итак, пусть дан граф G и его секущее множество S' размера $k+1$ и необходимо проверить, есть ли секущее множество S размера k.
- Расщепимся на 3^{k+1} случай: каждая вершина множества S' должна принадлежать либо S, либо одной из двух долей.

Решение compression version

- Итак, пусть дан граф G и его секущее множество S' размера $k+1$ и необходимо проверить, есть ли секущее множество S размера k.
- Расщепимся на 3^{k+1} случай: каждая вершина множества S' должна принадлежать либо S, либо одной из двух долей.
- В каждом из случаев у нас некоторые вершины графа уже принадлежат искомому секущему множеству S, некоторые уже покрашены в белый цвет, некоторые — в черный. Ясно, что в искомой раскраске графа $G \setminus S$ все соседи уже покрашенных черных вершин должны быть белыми, а все соседи уже покрашенных белых — черными.

Решение compression version

- Итак, пусть дан граф G и его секущее множество S' размера $k+1$ и необходимо проверить, есть ли секущее множество S размера k.
- Расщепимся на 3^{k+1} случай: каждая вершина множества S' должна принадлежать либо S, либо одной из двух долей.
- В каждом из случаев у нас некоторые вершины графа уже принадлежат искомому секущему множеству S, некоторые уже покрашены в белый цвет, некоторые — в черный. Ясно, что в искомой раскраске графа $G \setminus S$ все соседи уже покрашенных черных вершин должны быть белыми, а все соседи уже покрашенных белых — черными.
- Достаточно показать, что следующая задача $\in$ FPT. Дан двудольный граф G и непересекающиеся подмножества его вершин B и W. Необходимо проверить, существует ли такое подмножество его вершин S размера не более k, такое что $G \setminus S$ можно покрасить в два цвета, причем вершины множеств B и W будут покрашены, соответственно, в черный и белый цвета.

Вспомогательная задача

Вспомогательная задача

- Дан двудольный граф G и непересекающиеся подмножества его вершин B и W. Необходимо проверить, существует ли такое подмножество его вершин S размера не более k, такое что $G \setminus S$ можно покрасить в два цвета, причем вершины множеств B и W будут покрашены, соответственно, в черный и белый цвета.

Вспомогательная задача

- Дан двудольный граф G и непересекающиеся подмножества его вершин B и W. Необходимо проверить, существует ли такое подмножество его вершин S размера не более k, такое что $G \setminus S$ можно покрасить в два цвета, причем вершины множеств B и W будут покрашены, соответственно, в черный и белый цвета.
- Найдем какую-нибудь раскраску (B_0, W_0) графа G в два цвета.

Вспомогательная задача

- Дан двудольный граф G и непересекающиеся подмножества его вершин B и W. Необходимо проверить, существует ли такое подмножество его вершин S размера не более k, такое что $G \setminus S$ можно покрасить в два цвета, причем вершины множеств B и W будут покрашены, соответственно, в черный и белый цвета.
- Найдем какую-нибудь раскраску (B_0, W_0) графа G в два цвета.
- Вершины множества $C = (B_0 \cap W) \cup (W_0 \cap B)$ должны изменить цвет, а вершины множества $R = (B_0 \cap B) \cup (W_0 \cap W)$ — сохранить.

Вспомогательная задача

- Дан двудольный граф G и непересекающиеся подмножества его вершин B и W. Необходимо проверить, существует ли такое подмножество его вершин S размера не более k, такое что $G \setminus S$ можно покрасить в два цвета, причем вершины множеств B и W будут покрашены, соответственно, в черный и белый цвета.
- Найдем какую-нибудь раскраску (B_0, W_0) графа G в два цвета.
- Вершины множества $C = (B_0 \cap W) \cup (W_0 \cap B)$ должны изменить цвет, а вершины множества $R = (B_0 \cap B) \cup (W_0 \cap W)$ — сохранить.

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство леммы

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство леммы

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство

Доказательство леммы

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство

- $\Rightarrow$: в 2-раскраске G каждая вершина либо меняет, либо сохраняет цвет, причем со смежными вершинами происходит одно и то же.

Доказательство леммы

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство

- $\Rightarrow$: в 2-раскраске G каждая вершина либо меняет, либо сохраняет цвет, причем со смежными вершинами происходит одно и то же.
- $\Leftarrow$: Изменим цвет компонент графа $G \setminus S$, содержащих вершины множества C. $\square$

Доказательство леммы

Лемма

$G \setminus S$ имеет необходимую раскраску тогда и только тогда, когда S разделяет множества C и R (то есть ни одна компонента $G \setminus S$ не содержит одновременно вершины и из S, и из R).

Доказательство

- $\Rightarrow$: в 2-раскраске G каждая вершина либо меняет, либо сохраняет цвет, причем со смежными вершинами происходит одно и то же.
- $\Leftarrow$: Изменим цвет компонент графа $G \setminus S$, содержащих вершины множества C. $\square$

Алгоритм для нахождения множества S

Множество S может быть найдено за k итераций алгоритма Форда-Фалкерсона.

Спасибо за внимание!