

Алгоритмы и структуры данных

Лекция 6: Жадные алгоритмы

А. Куликов

Академия современного программирования

План лекции

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

План лекции

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

Общая идея

Общая идея

- На каждом шаге жадный алгоритм делает локально оптимальный выбор.
- Решение, найденное таким образом, не всегда оказывается оптимальным, но в ряде случаев все-таки оказывается.
- Общеизвестный пример — алгоритм для построения минимального покрывающего дерева: на каждом шаге берется самое легкое из возможных ребер.

Задача о выборе заявок

Определение

- Даны n пар чисел (s_i, f_i) , где $s_i < f_i$ обозначает время начала занятия, а f_i — конец.
- Говорим, что заявки (s_i, f_i) и (s_j, f_j) **совместны**, если интервалы $[s_i, f_i)$ и $[s_j, f_j)$ не пересекаются: $f_i \leq s_j$ или $f_j \leq s_i$.
- **Задача о выборе заявок** (activity-selection problem) заключается в выборе максимального количества совместных друг с другом заявок.

Замечание

Будем считать, что заявки отсортированы в порядке возрастания времени окончания: $f_1 \leq f_2 \leq \dots \leq f_n$.

Алгоритм

GREEDY-ACTIVITY-SELECTOR(s, f)

```
1  $n \leftarrow \text{length}[s]$ 
2  $A \leftarrow \{1\}$ 
3  $j \leftarrow 1$ 
4 for  $i \leftarrow 2$  to  $n$ 
5     do if  $s_i \geq f_j$ 
6         then  $A \leftarrow A \cup \{i\}$ 
7              $j \leftarrow i$ 
8 return  $A$ 
```

Анализ алгоритма

Анализ алгоритма

- Время работы есть $\Theta(n)$ (при условии, что заявки отсортированы!).
- Существует оптимальное решение, содержащее заявку 1: если в некотором оптимальном множестве заявка 1 не содержится, то первую заявку (то есть заявку с самым ранним временем окончания) этого множества можно заменить на заявку 1. При такой операции совместность не нарушится, а количество заявок останется прежним.
- Итак, мы ищем решение, содержащее заявку 1. Значит, можно выкинуть все заявки, несовместные с ней.
- Получаем подзадачу с меньшим количеством заявок.

Когда применимы жадные алгоритмы?

Две отличительные особенности жадных алгоритмов

- 1 **Принцип жадного выбора:** последовательность локально оптимальных (жадных) выборов дает глобально оптимальное решение. Для установления этого свойства, как правило, нужно показать, что жадный выбор согласован с некоторым оптимальным решением и что после выбора получается аналогичная подзадача.
- 2 **Свойство оптимальности для подзадач:** оптимальное решение для задачи содержит оптимальные решения для подзадач.

Жадный алгоритм или динамическое программирование?

Определение

В **непрерывной задаче о рюкзаке** (fractional knapsack problem), в отличие от общей задачи о рюкзаке, в рюкзак разрешать класть части предметов.

Замечания

- Нетрудно видеть, что жадный алгоритм находит оптимальное решение для непрерывной задачи о рюкзаке: на каждом шаге добавляем максимальное количество предмета максимальной удельной стоимости (стоимость/объём).
- Нетрудно также убедиться в том, что аналогичный алгоритм для общей задачи может и не найти оптимального решения: сразу положив в рюкзак самый дорогой предмет, мы можем потерять возможность полностью заполнить рюкзак.

Жадный алгоритм или динамическое программирование?

Замечания

- И так, в первом случае выполняется принцип жадного выбора, во втором — нет. Поэтому непрерывная задача решается жадным алгоритмом, общая — динамическим программированием.
- Слегка модифицировав жадный алгоритм для общей задачи о рюкзаке, можно получить 2-приближенный алгоритм.

План лекции

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

Коды Хаффмена

Коды Хаффмена

- Рассмотрим строчку длины 130, состоящую только из символов A, B, C, D . Более того, допустим, что частота каждого символа известна:

A	B	C	D
70	3	20	37

- Наша задача состоит в том, чтобы каждому символу присвоить битовый код так, чтобы соответственно закодированная строчка имела как можно меньшую длину.
- Один из вариантов кодирования: $A = 00, B = 01, C = 10, D = 11$. В коде тогда будет 260 бит.
- Интуитивно ясно, что хочется придумать кодирование, при котором символ A кодировался бы одним битом (возможно, ценой того, что B бы кодировался тремя).

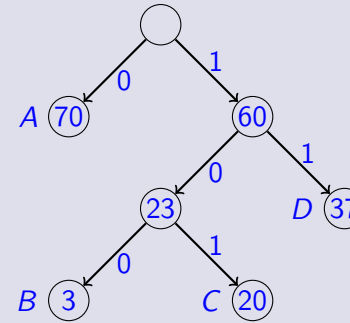
Коды Хаффмена (продолжение)

Коды Хаффмена

- Однако при кодировании символов последовательностями битов разной длины может возникнуть проблема декодировки.
- Одним из способов решения такой проблемы является **префиксное кодирование**.
- При таком кодировании ни для каких двух символов код одного не является префиксом кода другого.
- Каждое такое кодирование может быть представлено полным (у каждой вершины либо ноль, либо два сына) бинарным деревом, на ребрах которого стоят 0 и 1, а в листьях — кодируемые символы.

Пример кодирования

Дерево



Кодирование

A	B	C	D
70	3	20	37
0	100	101	11

Закодированная строка содержит 213 битов.

Оптимальное дерево

Оптимальное дерево

- Наша задача состоит в нахождении такого полного бинарного дерева, листья которого помечены символами и которое минимизирует **стоимость дерева**, определяемую как

$$\sum_{i=1}^n f_i \cdot (\text{глубина } i\text{-го символа в дереве}).$$

- Определим частоту внутренней вершины дерева как сумму частот её сыновей. Нетрудно видеть, что при таком определении частота каждой вершины дерева равняется просто количеству посещений этой вершины при кодировании или декодировании.
- Стоимость дерева тогда равняется сумме частот всех вершин дерева кроме корня.

Идея жадного построения дерева

Идея жадного построения дерева

- Ясно, что два самых редких символа должны висеть в самом низу оптимального дерева.
- НУО, f_1 , f_2 суть минимальные частоты.
- Стоимость оптимального дерева для частот f_1, \dots, f_n , в котором f_1 и f_2 являются листьями-братьями, равняется сумме $(f_1 + f_2)$ и стоимости оптимального дерева для частот $(f_1 + f_2), \dots, f_n$.

Алгоритм

HUFFMAN(f)

```
1  ▷ Вход: массив частот  $f[1..n]$ 
2  ▷ Выход: кодирующее дерево с  $n$  листьями
3  создать очередь с приоритетами  $H$ , упорядоченную по  $f$ 
4  for  $i \leftarrow 1$  to  $n$ 
5      do Insert( $H, i$ )
6  for  $k \leftarrow n+1$  to  $2n-1$ 
7      do  $i \leftarrow$  Extract-Min( $H$ )
8           $j \leftarrow$  Extract-Min( $H$ )
9          создать вершину с номером  $k$  и сыновьями  $i, j$ 
10          $f[k] \leftarrow f[i] + f[j]$ 
11         Insert( $H, k$ )
```

План лекции

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

Задача о покрытии множествами

Определение

- Дано множество $U = \{u_1, \dots, u_n\}$ и семейство его подмножеств $\mathcal{F} = \{S_1, \dots, S_k\}$, $S_i \subseteq U$.
- В сумме все подмножества покрывают U : $U = \bigcup_{S \in \mathcal{F}} S$.
- **Задача о покрытии множествами** (set cover problem) заключается в нахождении минимального набора подмножеств, покрывающего все множество U .

Замечания

В общем случае задача является NP-трудной. Мы построим приближенный алгоритм, основанный на жадной эвристике.

Алгоритм и анализ

Алгоритм и анализ

- Алгоритм на каждом шаге выбирает множество, покрывающее максимальное число все еще не покрытых элементов.
- Несложно построить пример, на котором такой жадный алгоритм выдает неоптимальное решение.
- Можно, однако, показать, что если оптимальное покрытие содержит m множеств, то наш жадный алгоритм найдет решение, содержащее не более $m \ln n$ множеств.
- Обозначим через n_t количество непокрытых элементов после t шагов.
- Поскольку все эти элементы покрываются m множествами из оптимального покрытия, найдется множество, покрывающее хотя бы n_t/m из них.

Анализ

- Значит, жадная эвристика гарантирует, что

$$n_{t+1} \leq n_t - \frac{n_t}{m} = n_t \left(1 - \frac{1}{m}\right).$$

-

$$n_t \leq n_0 \left(1 - \frac{1}{m}\right)^t < n_0 \left(e^{-1/m}\right)^t = ne^{-t/m}.$$

- При $t = m \ln n$, таким образом, n_t будет строго меньше 1.

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

Задача о вершинном покрытии

Определение

Задача о вершинном покрытии (vertex cover problem) заключается в нахождении по данному графу $G = (V, E)$ такого минимального его подмножества вершин V' , что для любого ребра $(u, v) \in E$ хотя бы одна из вершин u и v содержится в V' .

Простой жадный алгоритм

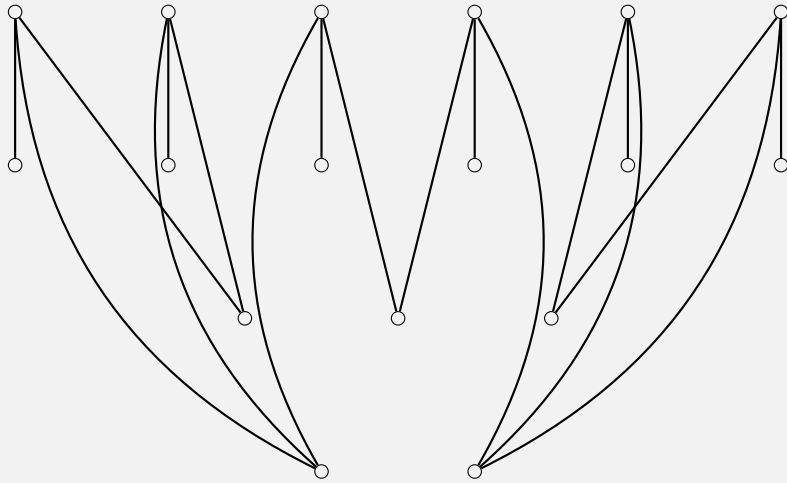
- Рассмотрим следующий жадный алгоритм: на каждом шаге выбираем вершину максимальной степени и выкидываем из графа все покрытые ребра.
- Мы покажем, что для любой константы c найдется граф, для которого этот алгоритм выдаст покрытие, которое по размеру будет более чем в c раз хуже оптимального.

Идея конструкции

Идея конструкции

- Рассматриваем двудольный граф на долях U и V .
- $|U| = n!$.
- $V = V_1 \cup V_2 \cup \dots \cup V_n$, $|V_i| = n!/i$.
- Каждая вершина $u \in U$ имеет ровно одного соседа в группе V_i .
- Таким образом, $\deg(v) = i$ для любой $v \in V_i$.
- Ясно, что U является вершинным покрытием.
- Жадный же алгоритм может последовательно выбрать сначала все вершины из V_n , потом все вершины из V_{n-1} и т.д.
- Полученное покрытие будет содержать $n!(1/n + 1/(n-1) + \dots + 1)$ вершин.

Пример для $n = 6$



План лекции

- 1 Введение
 - Что такое жадный алгоритм?
 - Пример: задача о выборе заявок
 - Когда применимы жадные алгоритмы?
- 2 Коды Хаффмена
- 3 Покрытие множествами
- 4 Вершинное покрытие
- 5 Локальный поиск для SAT

Формула в КНФ

Определение

- **Пропозициональной или Булевой** (propositional, Boolean) переменной называется переменная, принимающая значения true (1) и false (0).
- **Литералом** (literal) называется Булева переменная x или ее отрицание $\neg x$.
- **Клозом** (clause) называется дизъюнкция конечного множества литералов, не содержащего одновременно переменной и ее отрицания.
- **k -клозом** (k -clause) называется клоз, содержащий ровно k литералов.
- **Формулой в конъюнктивной нормальной форме (КНФ)** (formula in conjunctive normal form, CNF) называется конъюнкция конечного множества клозов.

Задача выполнимости

Определение

Задача пропозициональной выполнимости (Boolean satisfiability problem, SAT): определить, выполнима ли данная формула в КНФ, то есть существует ли набор Булевых значений переменным формулы, выполняющий формулу. Такой набор называют **выполняющим** (satisfying assignment), а формулу, для которой такой набор существует, — **выполнимой** (satisfiable).

Пример

- $F_1 = (x \vee y \vee \neg z) \wedge (\neg x) \wedge (\neg y \vee z)$
 F_1 выполнима: $x = 0, y = 1, z = 1$
- $F_2 = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$
 F_2 невыполнима, но три клоза можно выполнить

Жадный локальный поиск

GSAT(F , MAXFLIPS, MAXTRIES)

```
1 for  $i \leftarrow 1$  to MAXTRIES
2   do выбрать случайный набор  $I$ 
3   for  $j \leftarrow 1$  to MAXFLIPS
4     do если  $I$  выполняет  $F$ , вернуть  $I$ 
5     выбрать переменную  $v$ , при изменении которой
        выполняется максимальное возможное
        количество клозов  $F$ 
6     изменить значение переменной  $v$  в наборе  $I$ 
7 return "не знаю"
```

Сложная формула для алгоритма

первый тип клозов $(x_1 \vee \neg x_2), (x_2 \vee \neg x_3), \dots, (x_n \vee \neg x_1)$
второй тип клозов $(x_1 \vee x_2)$
третий тип клозов $(x_1 \vee \neg x_2 \vee \dots \vee \neg x_{n-3} \vee \neg x_{n-2} \vee \neg x_{n-1})$
 $(x_1 \vee \neg x_2 \vee \dots \vee \neg x_{n-3} \vee \neg x_{n-1} \vee \neg x_n)$
 $(x_1 \vee \neg x_2 \vee \dots \vee \neg x_{n-3} \vee \neg x_n \vee \neg x_{n-2})$
 $(x_2 \vee \neg x_3 \vee \dots \vee \neg x_{n-2} \vee \neg x_{n-1} \vee \neg x_n)$
 $(x_2 \vee \neg x_3 \vee \dots \vee \neg x_{n-2} \vee \neg x_n \vee \neg x_1)$
 $(x_2 \vee \neg x_3 \vee \dots \vee \neg x_{n-2} \vee \neg x_1 \vee \neg x_{n-1})$
...
 $(x_n \vee \neg x_1 \vee \dots \vee \neg x_{n-4} \vee \neg x_{n-3} \vee \neg x_{n-2})$
 $(x_n \vee \neg x_1 \vee \dots \vee \neg x_{n-4} \vee \neg x_{n-2} \vee \neg x_{n-1})$
 $(x_n \vee \neg x_1 \vee \dots \vee \neg x_{n-4} \vee \neg x_{n-1} \vee \neg x_{n-3})$

Идеи доказательства

Идеи доказательства

- Выполняющий набор у формулы всего один.
- Если текущий набор алгоритма находится на расстоянии 2 от выполняющего, то следующим шагом алгоритм обязательно сделает неверный ход.
- Таким образом, если случайно выбранный набор не попадет в множество наборов, находящихся на расстоянии не более 1 от выполняющего, то жадный алгоритм за одно блуждание его не найдёт.
- Поскольку всего различных наборов 2^n , а упомянутых — всего $n + 1$, жадный алгоритм находит выполняющий набор с экспоненциально маленькой вероятностью.

Формальное доказательство

Теорема

Пусть $S = \{x_1 = 1, x_2 = 1, \dots, x_n = 1\}$ — единственный выполняющий набор формулы F , I — набор, такой что $d(S, I) = 2$. Тогда для любой переменной v , такой что $d(S, I^v) = 1$, I^v выполняет меньше клозов, чем I .

Доказательство

- Пусть I отличается от S значениями переменных $v = x_i$ и x_j .
- Ясно, что I^v выполняет клоз $(x_1 \vee x_2)$, $3n - 3$ клоза третьего типа и $n - 1$ клозов первого типа. Всего $4n - 3$ клоза, таким образом.

Формальное доказательство (продолжение)

Доказательство

- Рассмотрим три случая:
 - 1 $|(i - j) \pmod n| \neq 1$. Тогда I выполняет $n - 2$ клона первого типа, клон $(x_1 \vee x_2)$ и $3n - 1$ или $3n$ клонов третьего типа. Всего не менее $4n - 2$.
 - 2 $|(i - j) \pmod n| = 1$, но $\{i, j\} \neq \{1, 2\}$. Тогда $n - 1$ клона первого типа, клон $(x_1 \vee x_2)$ и $3n - 1$ клонов третьего типа. Итого $4n - 1$.
 - 3 $\{i, j\} = \{1, 2\}$. $n - 1$ клон первого типа и $3n - 1$ клона третьего типа. $4n - 2$ клона. \square

Завершающая теорема

Теорема

Пусть I — набор, такой что $d(S, I) = 2$. Тогда найдется переменная v , такая что $d(S, I^v) = 3$ и I^v выполняет столько же клозов, сколько и I .

Доказательство

- Пусть I отличается от S значениями переменных x_i и x_j .
- Выберем v так, чтобы
 - 1 $v \neq x_1, x_2$;
 - 2 выполнено ровно одно из условий: $v = x_{j+1}$, $v = x_{j-1}$, $v = x_{i+1}$, $v = x_{i-1}$ (это возможно при $n \geq 6$).
- Нетрудно видеть, что тогда I^v выполняет те же клозы второго и третьего типов, что и I , и столько же клозов первого типа, что и I . \square

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Две отличительные особенности жадного алгоритма: принцип жадного выбора и свойство оптимальности для подзадач.
- Свойство оптимальности для подзадач выполняется и для динамического программирования.
- Жадная стратегия в ряде случаев находит решение, которое не сильно хуже оптимального.