Адрианов Н.М.
Иванов А.Б.

Алгоритмы и структуры данных

Жадные алгоритмы (greedy)

Задача о расписании

- Процессор (исполнитель) – может выполнять 1 задачу в каждый момент времени.
- Список задач P_1, P_2, ... P_n.
- Для каждой задачи заданы длительность l_i и приоритет w_i.

Пусть выбран порядок выполнения задач P_{m_1}, P_{m_2}, P_{m_3}, ...
Время окончания выполнения задачи = сумма длительностей и задачи и всех задач перед ней:

$$C_{m_k} = \sum_{i=1}^{k} l_{m_i}$$

Требуется: найти порядок выполнения задач, при котором величина

$$\sum_{i=1}^{n} w_i C_i$$

будет минимальна.

Задача о расписании: жадный алгоритм

Утв. Упорядочим задачи по *убыванию* величины w_i/l_i.
Этот порядок будет оптимальным.

Док-во. Предположим, что оптимальный порядок другой. Тогда в нем найдется пара задач, идущих подряд

$$\ldots, P_i, P_j, \ldots$$

для которых

$$w_i/l_i < w_j/l_j.$$

Поменяем порядок выполнения задач P_i и P_j: $\ldots$

Задача о кэшировании (cache)

Память компьютера разбита на страницы. Чтобы процессор получил доступ к данным, надо зачитать их в кэш. Читается страница целиком. В кэш помещается m страниц.

Дано: последовательность запросов к страницам.

Требуется: определить оптимальный сценарий сброса страниц из кэша, при котором количество промахов (cache miss) будет минимально.

Пример: $m = 3$. Последовательность запросов

$$a,\ b,\ c,\ a,\ d,\ d,\ c,\ d,\ a,\ b$$

Задача о кэшировании: жадный алгоритм

Теорема (Bélády, 1966). Выкидываем из кэша ту страницу, которая понадобится в наиболее далеком будущем. Это оптимальная стратегия.

Доказательство использует похожую идею с перестановкой, как в задаче о расписании, но рассуждение более сложное.

Выбор непересекающихся интервалов

Дано: набор (возможно пересекающихся) временных интервалов (s_i, f_i).

Требуется: выбрать максимально возможное количество непересекащихся интервалов.

Выбор непересекающихся интервалов

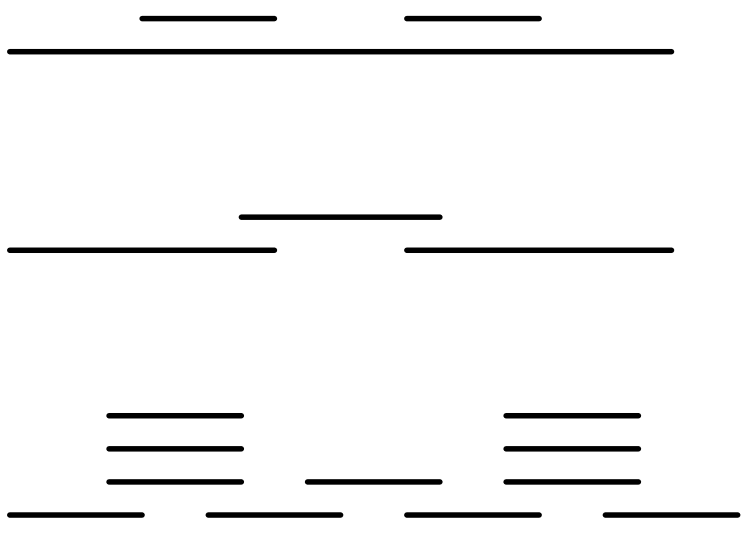
Идеи, как организовать жадный алгоритм: выбираем интервал

1. с минимальной начальной точкой s_i
2. минимальной длины $f_i - s_i$
3. с минимальным количеством пересечений

Выбор непересекающихся интервалов

Идеи, как организовать жадный алгоритм: выбираем интервал

1. с минимальной начальной точкой s_i
2. минимальной длины $f_i - s_i$
3. с минимальным количеством пересечений



Выбор непересекающихся интервалов

Выбираем интервал с самым ранним временем окончания f_i. Выкидываем все интервалы, пересекающиеся с ним. Повторяем.

Утв. Это оптимальная стратегия.

Док-во. Пусть I_0 – интервал с самым ранним временем окончания. Возьмем оптимальный набор интервалов. Если I_0 в него не входит – заменим самый левый интервал в наборе на I_0.

Жадные алгоритмы на графах

- алгоритм Дейкстры (кратчайшие пути)
- алгоритм Прима (минимальное остовное дерево)
- алгоритм Крускала (минимальное остовное дерево)

Бинарное кодирование

Имеется алфавит Σ

Для каждой буквы $\alpha \in \Sigma$ определяем код – последовательность бит (коды могут быть разной длины)

$$c : \Sigma \to \{0, 1\}^*$$

Единственный ли способ раскодировать слово?

Безпрефиксные коды (prefix-free)

Код называется безпрефиксным, если не существует двух букв таких, что код одной является началом кода другой.

$$\nexists\, \alpha,\, \beta \in \Sigma :\ c(\alpha) \text{ является началом } c(\beta)$$

Безпрефиксный код раскодируется однозначно.

Если коды букв алфавита изобразить в виде бинарного дерева – буквы будут стоять только в листьях дерева.

Оптимальный беспрефиксный код

Дано: для каждого символа $\alpha \in \Sigma$ известна частота $p(\alpha)$ появления его в тексте.

Требуется: найти беспрефиксный код, для которого сумма

$$\sum_{\alpha \in \Sigma} p(\alpha)|c(\alpha)|$$

минимальна ($|c(a)|$ – длина кодового слова для символа α).

Алгоритм Хаффмана

Алгоритм Хаффмана

```
procedure Huffman( chars, p ):
  // chars - массив символов (алфавит)
  // p - массив частот символов
  // (кол-во символов = кол-во частот = n)
  Q = priority queue
  // создадим n листьев
  for i = 0 to n-1:
      node = вершина(
          ch = chars[i],
          freq = p[i])
      Q.add(node)

  // создадим еще n-1 внутренних вершин
  for k = n to 2n-2:
      node1 = Q.deleteMin()
      node2 = Q.deleteMin()
      node = вершина(
          freq  = node1.freq + node.freq,
          left  = node1,
          right = node2)
      Q.add(node)
  end
end
```

```
class Node {
  char ch;
  int  freq;
  Node left;
  Node right;

  ...
}
```

Задача о рюкзаке

Дано:

- Рюкзак максимальной вместимости W (по весу)
- n предметов, для которых задан вес w_i и ценность v_i

Требуется: Выбрать предметы общего веса $\leq W$ и максимальной суммарной ценности.

Задача о рюкзаке: жадный алгоритм

Предположим, что можно брать дробные части предметов. (например, есть песок золотой, серебрянный и медный и можно насыпать любое количество из имеющегося в наличии)

Очевидно: надо насыпать в рюкзак самый ценный (золотой). Если в рюкзаке осталось место – следующий по ценности (серебрянный) и т.д.

Жадный алгоритм дает решение с ценностью

$$v_{greedy-fractional} = v_{optimal-fractional}$$

Причем

$$v_{optimal-fractional} \geq v_{optimal}$$

Задача о рюкзаке: жадный алгоритм

Алгоритм (не очень хороший).

1. Упорядочим предметы
   по убыванию относительной ценности v_i/w_i.
   $$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \frac{v_3}{w_3} \geq \dots$$
2. Кладем в рюкзак предметы $1, 2, \dots k$, пока они помещаются.

Пример: $W = 1000$

$$\begin{array}{ll} v_1 = 2 & w_1 = 1 \\ v_2 = 1000 & w_2 = 1000 \end{array}$$

$$v_{optimal} = 1000$$

$$v_{greedy} = 2$$

Задача о рюкзаке: жадный алгоритм

Алгоритм (хороший).

1. Упорядочим предметы
   по убыванию относительной ценности v_i/w_i.
   $$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \frac{v_3}{w_3} \geq \dots$$
2. Кладем в рюкзак предметы $1, 2, \dots k$, пока они помещаются.
   Получили некоторое решение с ценностью
   $v_{greedy} = v_1 + \dots + v_k$.
3. Если есть предмет (w_{max}, v_{max}): $w_{max} \leq W$, $v_{greedy} < v_{max}$, то вместо решения, полученного на шаге 2 берем решение, состоящее из одного этого предмета.

Анализ эффективности жадного алгоритма

Утв. Ценность решения по «хорошему» жадному алгоритму не меньше $\frac{1}{2}$ ценности оптимального решения ($\frac{1}{2}$-аппроксимация).

Пример: $W = 1000$

$$\begin{aligned} v_1 &= 502 \quad w_1 = 501 \\ v_2 &= 500 \quad w_2 = 500 \\ v_3 &= 500 \quad w_3 = 500 \end{aligned}$$

$$v_{optimal} = 1000$$

$$v_{greedy-good} = 502$$

Анализ эффективности жадного алгоритма

Утв. Ценность решения по «хорошему» жадному алгоритму не меньше $\frac{1}{2}$ ценности оптимального решения ($\frac{1}{2}$-аппроксимация).

Доказательство:

$$v_{greedy} = \sum_{i=1}^{k} v_i$$

$$v_{greedy-good} = \max\left(\sum_{i=1}^{k} v_i,\ v_{max}\right)$$

$$v_{greedy-good} \geq \sum_{i=1}^{k} v_i$$

$$v_{greedy-good} \geq v_{max} \geq v_{k+1}$$

$$2 \cdot v_{greedy-good} \geq \sum_{i=1}^{k} v_i + v_{k+1} \geq v_{greedy-fractional} \geq v_{optimal}$$

Покрытие множествами

Дано: множество B и его подмножества $S_1, \dots S_m \subseteq B$.

Требуется: найти минимальный набор множеств $S_{i_1}, \dots S_{i_k}$, которые покрывают все B.

Покрытие множествами: жадный алгоритм

Критерий: выбираем S_i с максимальным числом непокрытых элементов.

Утв. Пусть $|B| = n$ и оптимальное покрытие состоит из k элементов. Тогда жадный алгоритм даст решение из не более, чем $k \ln n$ множеств.

Пусть n_t – количество непокрытых элементов после шага t в жадном алгоритме. Они покрываются k подмножествами (оптимальным покрытием). Значит, существует подмножество, покрывающее не менее n_t/k элементов, следовательно, на следующем шаге жадного алгоритма останутся непокрытыми

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

Поскольку $n_0 = n$ и $1 - x \leq e^{-x}$, то

$$n_t \leq \left(1 - \frac{1}{k}\right)^t n_0 < e^{-t/k} n$$

При $t = k \ln n$ получаем

$$n_t < e^{-k \ln n/k} n = 1 \quad \Rightarrow \quad n_t = 0$$