Алгоритми та структури даних
Модуль 2: Методи розробки алгоритмів
Лекція 8. Метод грубої сили.

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Послідовний пошук і пошук підстрок методом грубої сили.
2. Задачі пошуку пари найближчих крапок і обчислення опуклої оболонки.
3. Вичерпний перебір.

Питання по попередній лекції

- *Що таке прямий хід рекурсії?*
- *Що таке база рекурсії?*
- *Які алгоритми сортування вам відомі?*

Послідовний пошук.

АЛГОРИТМ $SequentialSearch2\,(A\,[0..n]\,,K)$

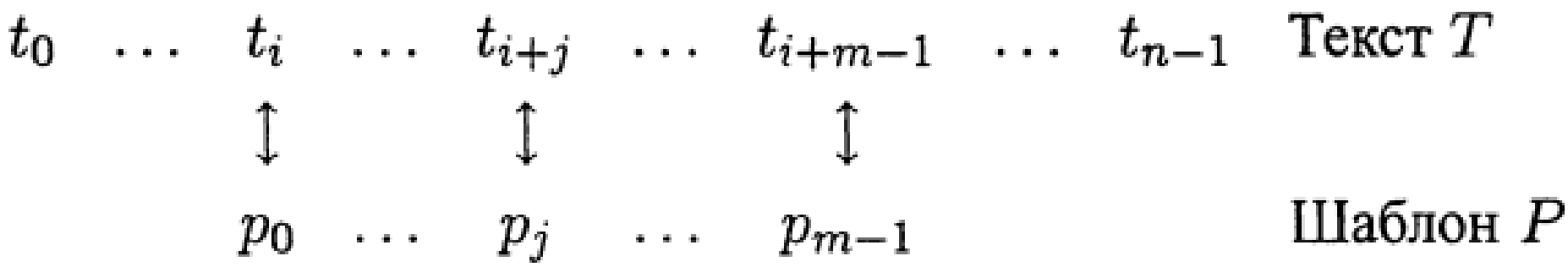
// Алгоритм реализует последовательный поиск
// с использованием ключа поиска в качестве ограничителя
// **Входные данные:** Массив A из n элементов и ключ поиска K
// **Выходные данные:** Позиция первого элемента массива
// $A[0..n-1]$, значение которого совпадает
// с K; если элемент не найден, возвращается
// значение -1

$A[n] \leftarrow K$
$i \leftarrow 0$
while $A[i] \neq K$ **do**
  $i \leftarrow i+1$
if $i < n$
  return i
else
  return -1

Пошук підстрок

Дано символьний рядок довжиною *n*, що називається текстом, і рядок довжиною *m* (m<=n), іменована шаблоном (pattern); потрібно знайти в тексті підстроки, що відповідає шаблону.

$$t_i = p_0, \ldots, t_{i+j} = p_j, \ldots, t_{i+m-1} = p_{m-1}:$$



Пошук підстрок

Алгоритм $BruteForceStringMatch\,(T\,[0..n-1]\,,P\,[0..m-1])$

// Алгоритм реализует поиск подстроки методом грубой силы
// **Входные данные:** массив $T[0..n-1]$ из n символов,
// представляющий текст;
// массив $P[0..m-1]$ из m символов,
// представляющий шаблон
// **Выходные данные:** позиция первого символа в тексте,
// с которой начинается первая искомая
// подстрока, соответствующая шаблону; если
// подстрока не найдена, возвращается -1

for $i \leftarrow 0$ **to** $n-m$ **do**
  $j \leftarrow 0$
  while $j < m$ **and** $P[j] = T[i+j]$ **do**
    $j \leftarrow j+1$
  if $j = m$
    return i
return -1

До якого класу ефективності відноситься алгоритм BruteForceStringMatch()???

Пошук найближчих точок

Задача пошуку пари найближчих точок полягає в тому, щоб у множині з n точок знайти дві, розташовані друг до друга ближче інших.

АЛГОРИТМ $BruteForceClosestPoints\,(P)$

// **Входные данные:** список P, состоящий из $n \geqslant 2$ точек
// $P_1 = (x_1, y_1), \ldots, P_n = (x_n, y_n)$
// **Выходные данные:** индексы $index1$ и $index2$ пары
// ближайших точек
$dmin \leftarrow \infty$
for $i \leftarrow 1$ **to** $n - 1$ **do**
  for $j \leftarrow i + 1$ **to** n **do**
    $d \leftarrow sqrt((x_i - x_j) * *2 +$ // $sqrt$ — функция вычисления
      $+(y_i - y_j) * *2)$ // квадратного корня,
    if $d < dmin$ // ** — возведение в степень
      $dmin \leftarrow d;\ index1 \leftarrow i;\ index2 \leftarrow j;$
return $index1, index2$

Часова ефективність алгоритму BruteForceClosestPoints:

$$C(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2 = 2 \sum_{i=1}^{n-1} (n-i) = 2\left[(n-1) + (n-2) + \cdots + 1\right] =$$
$$= (n-1)\, n \in \Theta\left(n^2\right).$$

Поняття опуклої оболонки

Визначення 1. Множина точок (кінцева або нескінченна) на площині називається опуклою, якщо для будь-яких двох точок P и Q, що належать даній множині, відрізок з кінцевими точками P и Q буде повністю належати цій же множині.

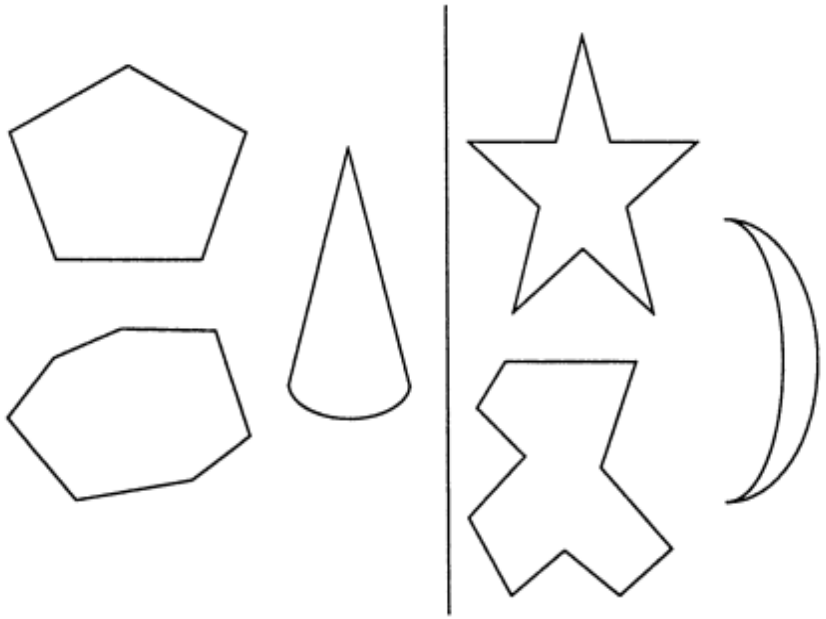
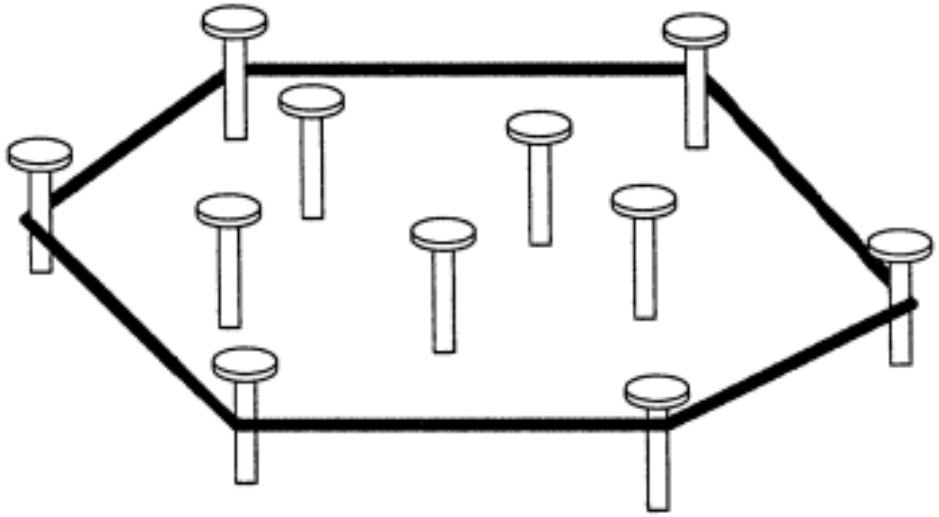


Рис. 3.4. *а*) Выпуклые множества. *б*) Множества, не являющиеся выпуклыми

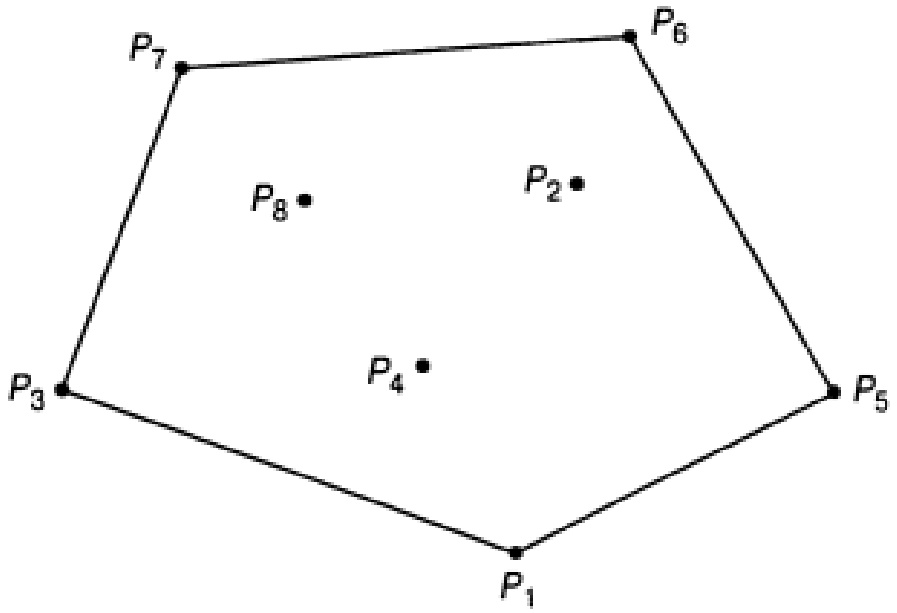
Поняття опуклої оболонки

Визначення 2. Опукла оболонка множини точок S являє собою найменшу опуклу множину, що містить S. ("Найменше" означає, що опукла оболонка S повинна бути підмножиною будь-якої опуклої множини, що містить S.)



Поняття опуклої оболонки

Теорема 1. Опукла оболонка довільної множини S, що складається з $n > 2$ точок (не належних одній прямій), являє собою опуклий багатокутник з вершинами в деяких точках S. (Якщо всі точки лежать на одній прямій, то багатокутник вироджується у відрізок прямої, дві кінцеві точки якого однаково належать множині S.)



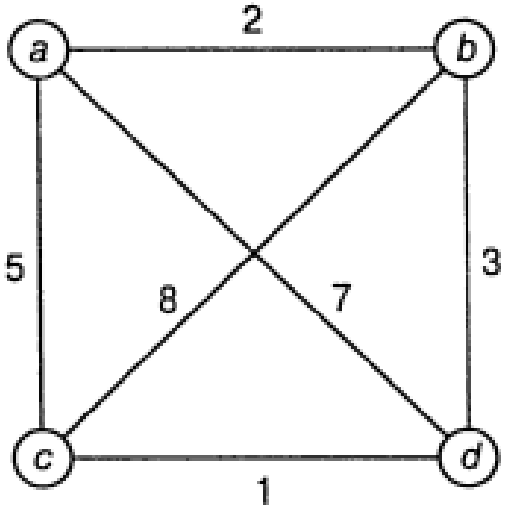
Вичерпний перебір

Вичерпний перебір (exhaustive search) являє собою підхід до комбінаторних задач із позиції грубої сили. Він припускає генерацію всіх можливих елементів з області визначення задачі, вибір тих з них, які задовольняють обмеженням, що накладається умовою задачі, і наступний пошук потрібного елемента (наприклад, оптимізуючого значення цільової функції задачі).

Задача комівояжера

Треба знайти такий найкоротший шлях по заданим *n* містах, щоб кожне місто відвідувалося тільки один раз і кінцевим пунктом виявилися місто, з якого починалася подорож.

Задача комівояжера



Путь	Длина	
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$	$l = 2 + 8 + 1 + 7 = 18$	
$a \rightarrow b \rightarrow d \rightarrow c \rightarrow a$	$l = 2 + 3 + 1 + 5 = 11$	Оптимален
$a \rightarrow c \rightarrow b \rightarrow d \rightarrow a$	$l = 5 + 8 + 3 + 7 = 23$	
$a \rightarrow c \rightarrow d \rightarrow b \rightarrow a$	$l = 5 + 1 + 3 + 2 = 11$	Оптимален
$a \rightarrow d \rightarrow b \rightarrow c \rightarrow a$	$l = 7 + 3 + 8 + 5 = 23$	
$a \rightarrow d \rightarrow c \rightarrow b \rightarrow a$	$l = 7 + 1 + 8 + 2 = 18$	

Задача про рюкзак

Дано n предметів вагою $w_1,\ldots,w_n$ й ціною $v_1,\ldots,v_n$, а також рюкзак, що витримує вага W. Потрібно знайти підмножину предметів, яку можна розмістити в рюкзаку, і яке має при цьому максимальну вартість.

Подмножество	Общий вес	Общая стоимость
$\varnothing$	0	0
$\{1\}$	7	42
$\{2\}$	3	12
$\{3\}$	4	40
$\{4\}$	5	25
$\{1,2\}$	10	36
$\{1,3\}$	11	Недопустим
$\{1,4\}$	12	Недопустим
$\{2,3\}$	7	52
$\{2,4\}$	8	37
$\{3,4\}$	9	65

Задача про призначення

Є *n* працівників, які повинні виконати *n* завдань, по одному завданню кожний (тобто кожному працівникові призначається тільки одне завдання, і кожне завдання призначається лише одній людині). Вартість виконання *i-м* працівником *j-го* завдання відома й дорівнює *C [i, j]* для всіх пар *i, j = 1,…,n*.

Задача полягає в наступному: треба розподілити завдання між працівниками таким чином, щоб вони були виконані з найменшою загальною вартістю.

Невеликий екземпляр даної задачі наведений у таблиці вартості виконання завдань *C [i, j]*:

Задача про призначення

	Задание 1	Задание 2	Задание 3	Задание 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

Задача про призначення

$$C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

<1, 2, 3, 4> стоимость = 9 + 4 + 1 + 4 = 18
<1, 2, 4, 3> стоимость = 9 + 4 + 8 + 9 = 30
<1, 3, 2, 4> стоимость = 9 + 3 + 8 + 4 = 24
<1, 3, 4, 2> стоимость = 9 + 3 + 8 + 6 = 26
<1, 4, 2, 3> стоимость = 9 + 7 + 8 + 9 = 33
<1, 4, 3, 2> стоимость = 9 + 7 + 1 + 6 = 23

и т.д.

Резюме

Резюме (що встигли за лекцію)

- Розглянули алгоритм послідовного пошуку.
- Розглянули алгоритм пошуку підстрок.
- Розглянули алгоритм пошуку пари найближчих точок.
- Розглянули задачі комівояжера, про рюкзак, про призначення.