

Построение и анализ алгоритмов

Лекция 4

Динамическое программирование

Оптимальные деревья поиска

Пример 3. Оптимальные деревья поиска

См. начало в Лекции 3.

См. также раздел 2.8

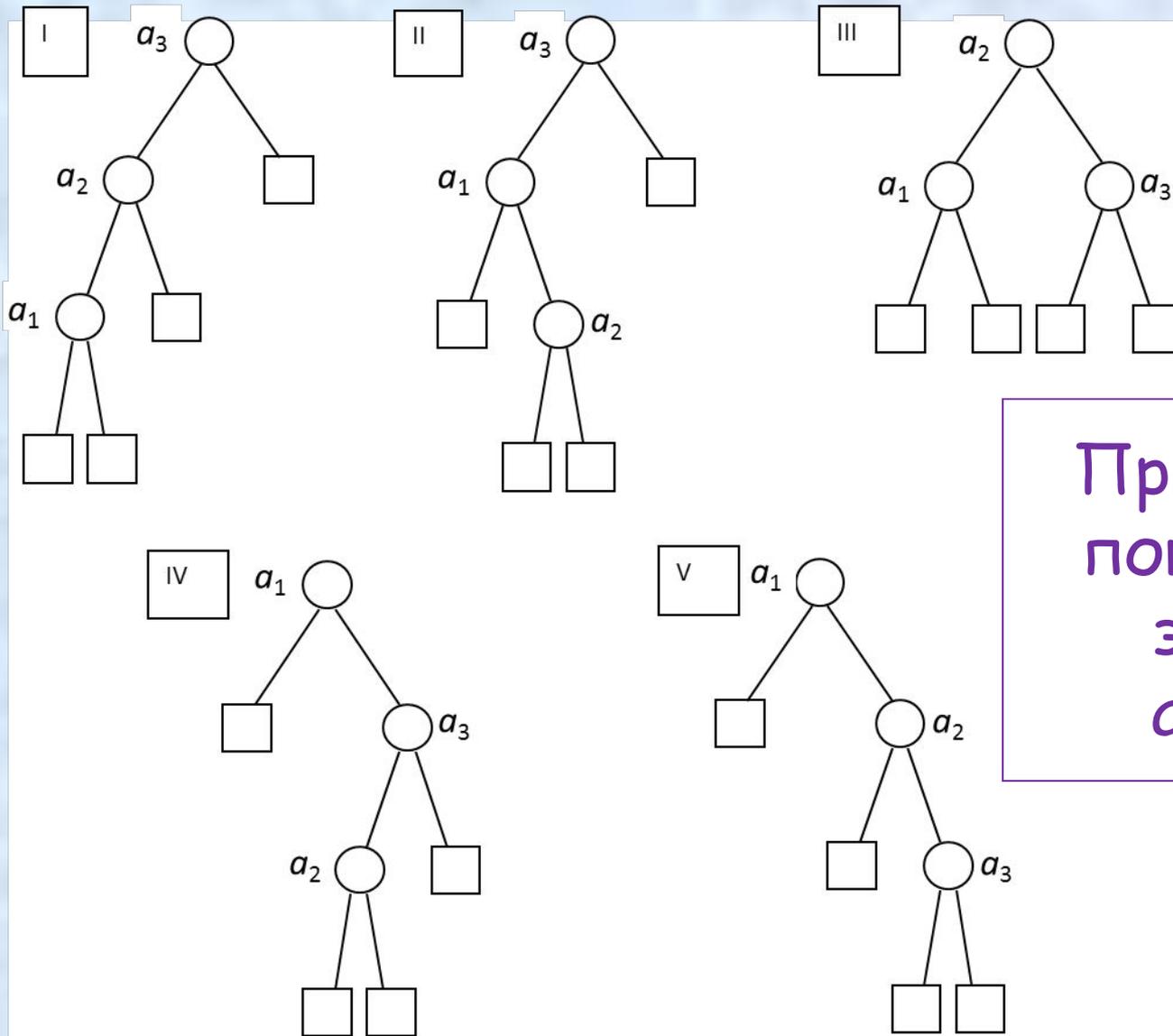
пособия «Деревья кодирования и поиска»

Оптимальные деревья поиска

Ранее при рассмотрении БДП, как правило, предполагалось, что для поиска различные ключи предъявляются с **равной вероятностью**.

Пусть теперь заранее известно, что некоторые ключи предъявляются чаще других.

Тогда расположение «частых» ключей ближе к корню дерева сократит время их поиска и, возможно, среднее время поиска (по разным предъявлениям ключей).



Пример дерева
поиска из трёх
элементов
 $a_1 < a_2 < a_3$.

Рис. 2.9. Все различные расширенные БДП из трёх элементов $a_1 < a_2 < a_3$

Заданы вероятности предъявления элемента x для поиска: $P(x = a_1) = \alpha$; $P(x = a_2) = \beta$; $P(x = a_3) = \gamma$.

Дерево	Стоимость	Варианты значений α , β и γ			
		$\alpha = 1/3$ $\beta = 1/3$ $\gamma = 1/3$	$\alpha = 3/6$ $\beta = 2/6$ $\gamma = 1/6$	$\alpha = 4/7$ $\beta = 2/7$ $\gamma = 1/7$	$\alpha = 5/8$ $\beta = 2/8$ $\gamma = 1/8$
		Значения стоимости при заданных α , β и γ			
I	$3\alpha + 2\beta + \gamma$	6/3	14/6	17/7	20/8
II	$2\alpha + 3\beta + \gamma$	6/3	13/6	15/7	17/8
III	$2\alpha + \beta + 2\gamma$	5/3	10/6	12/7	14/8
IV	$\alpha + 3\beta + 2\gamma$	6/3	11/6	12/7	13/8
V	$\alpha + 2\beta + 3\gamma$	6/3	10/6	11/7	12/8

Среднее (по всем предъявлениям x) число сравнений (стоимость) в случаях успешного поиска как функция переменных α , β и γ ,

Постановка задачи

Поиск будет осуществляться среди набора данных

$$a_1, a_2, \dots, a_{n-1}, a_n.$$

Пусть последовательность упорядочена:

$$a_1 < a_2 < \dots < a_{n-1} < a_n.$$

A_1, \dots, A_n - события, соответствующие вариантам успешных исходов поиска,

$$\text{т. е. } A_i: (x = a_i) \text{ для } i \in 1..n,$$

B_0, \dots, B_n - события, соответствующие вариантам неудачных исходов поиска,

$$\text{т. е. } B_i: (a_i < x < a_{i+1}) \text{ для } i \in 0..n.$$

Здесь для упрощения записи событий B_0 и B_n добавлены фиктивные элементы $a_0 = -\infty$ и $a_{n+1} = +\infty$, которые не должны использоваться в алгоритме.

Постановка задачи (продолжение)

Все эти $2n + 1$ событий (исходов поиска)

$$(A_i)_1^n \quad (B_i)_0^n$$

могут быть упорядочены:

$$B_0 < A_1 < B_1 < A_2 < \dots < B_{n-1} < A_n < B_n.$$

Заданы вероятности (или частоты) этих событий:

$$p_i = P(A_i) \text{ для } i \in 1..n, \text{ и } q_i = P(B_i) \text{ для } i \in 0..n.$$

$$\text{При этом } \sum_{i \in 1..n} p_i + \sum_{i \in 0..n} q_i = 1.$$

События A_i соответствуют внутренним узлам расширенного дерева поиска, а события B_i - внешним узлам (листьям) расширенного дерева поиска.

Постановка задачи (продолжение)

Тогда среднее число (математическое ожидание) сравнений при поиске можно записать в виде

$$C_{0,n} = \sum_{i=1}^n p_i (l(A_i) + 1) + \sum_{i=0}^n q_i l(B_i),$$

где $l(x)$ - уровень узла x (или длина пути от корня до узла x) в БДП.

Здесь уровень узла определён так, что $l(\text{корень}) = 0$.

Итак, задача состоит в том, чтобы по заданным весам

$$\{p_i\}_1^n \text{ и } \{q_i\}_0^n$$

построить БДП, минимизирующее значение $C_{0,n}$.

Постановка задачи (продолжение)

Такое дерево называют оптимальным БДП.

Есть ли сходство этой задачи с задачей построения оптимального префиксного кода ?

В чём сходство, в чём различие?

Ответ.

Очевидное решение поставленной задачи состоит в переборе всех структурно различных бинарных деревьев с n узлами и выборе дерева с наименьшей стоимостью $C_{0,n}$.

Однако поскольку (см. лекции про БДП) число b_n структурно различных бинарных деревьев с n узлами есть

$$b_n = \frac{4^n}{\sqrt{\pi n}^{3/2}}$$

, то этот способ вряд ли будет иметь практическую ценность.

Оказывается, приемлемое по количеству вычислений решение данной задачи может быть получено **методом динамического программирования.**

Конец повторения прошлой лекции

Построение оптимальных деревьев поиска

Дано:

1) набор элементов $a_1 < a_2 < \dots < a_{n-1} < a_n$.

2) набор весов

$$\{p_i\}_1^n \text{ и } \{q_i\}_0^n$$

$$1) \sum_{i \in 1..n} p_i + \sum_{i \in 0..n} q_i = 1.$$

Требуется: по заданным весам построить БДП, минимизирующее значение $C_{0,n}$.

$$C_{0,n} = \sum_{i=1}^n p_i (l(A_i) + 1) + \sum_{i=0}^n q_i l(B_i),$$

Идея

Пусть имеется оптимальное дерево.

Согласно принципу оптимальности, лежащему в основе метода динамического программирования, левое и правое поддеревья этого дерева в свою очередь также должны быть оптимальны.

T_{ij} - поддереву БДП из элементов a_{i+1}, \dots, a_j
 (при $0 \leq i \leq j \leq n$).

$$T_{ij} = \text{БДП}\{a_{i+1}, \dots, a_j\}$$

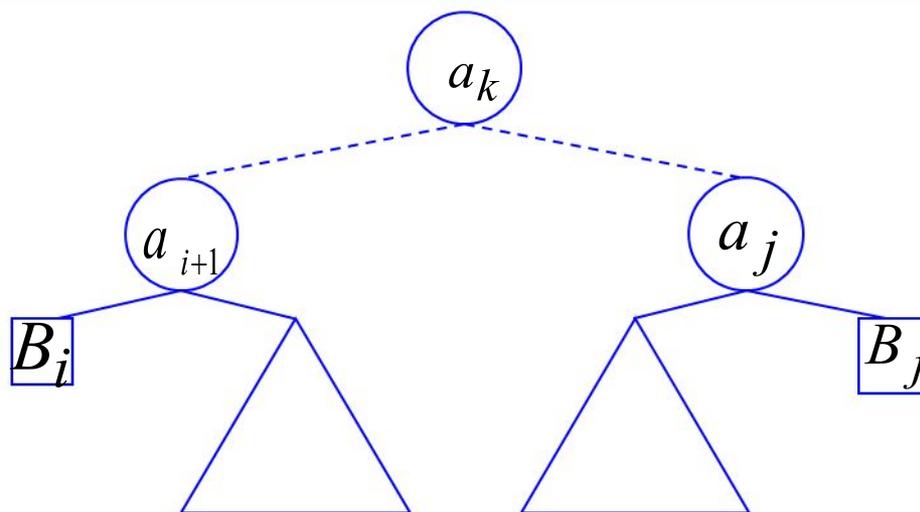


Рис. 2.10. Структура расширенного поддерева T_{ij}

корнем поддерева может быть любой из элементов , т. е.

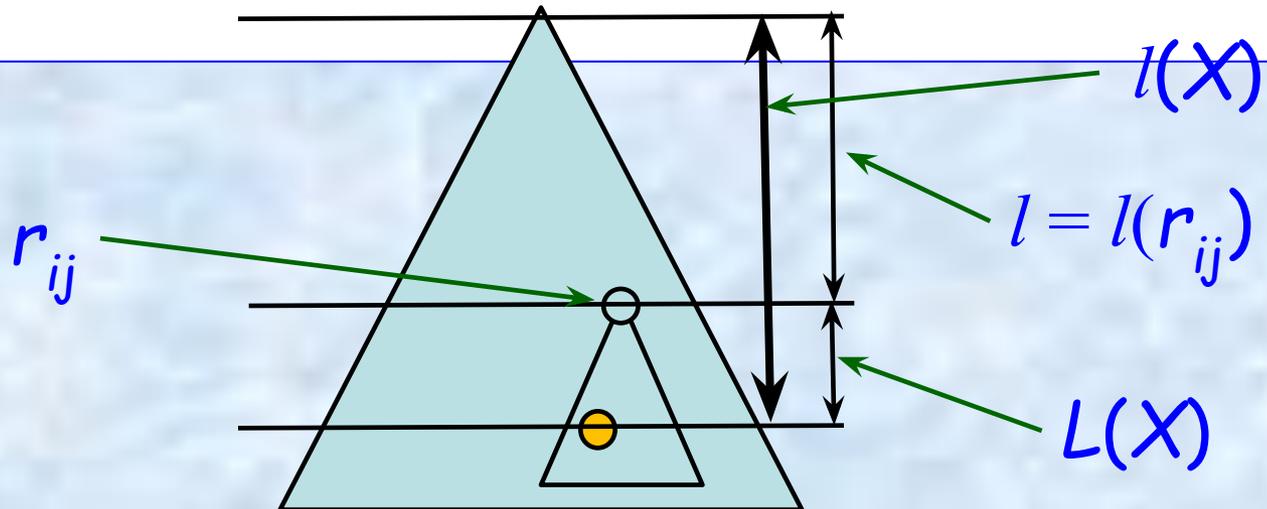
$$k \in i+1..j.$$

Пусть

$l = l(r_{ij})$ - уровень корня r_{ij} поддерева T_{ij} в дереве $T_{0,n}$

$L(X)$ - уровень узла, соответствующего событию X , в поддереве T_{ij} . ($L(r_{ij})=0$)

Тогда $l(X) = L(X) + l$, где $X \in \{B_i, A_{i+1}, \dots, B_j\}$.



Вклад поддереза T_{ij} в стоимость $C_{0,n}$

$$\begin{aligned} \sum_{k=i+1}^j p_k (l(A_k) + 1) + \sum_{k=i}^j q_k l(B_i) &= \sum_{k=i+1}^j p_k (L(A_k) + l + 1) + \sum_{k=i}^j q_k (L(B_i) + l) = \\ &= \sum_{k=i+1}^j p_k (L(A_k) + 1) + \sum_{k=i}^j q_k L(B_i) + \left(\sum_{k=i+1}^j p_k + \sum_{k=i}^j q_k \right) l = C_{ij} + w_{ij}l, \end{aligned}$$

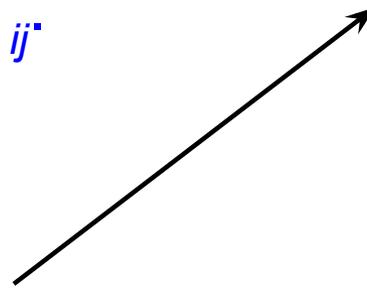
где

$$C_{ij} = \sum_{k=i+1}^j p_k (L(A_k) + 1) + \sum_{k=i}^j q_k L(B_i), \quad w_{ij} = \sum_{k=i+1}^j p_k + \sum_{k=i}^j q_k.$$

C_{ij} - стоимость поддереза T_{ij} .

w_{ij} - вес поддереза T_{ij} .

$$w_{ij} = w_{i,j-1} + (p_j + q_j)$$



Идея: структура дерева + принцип оптимальности

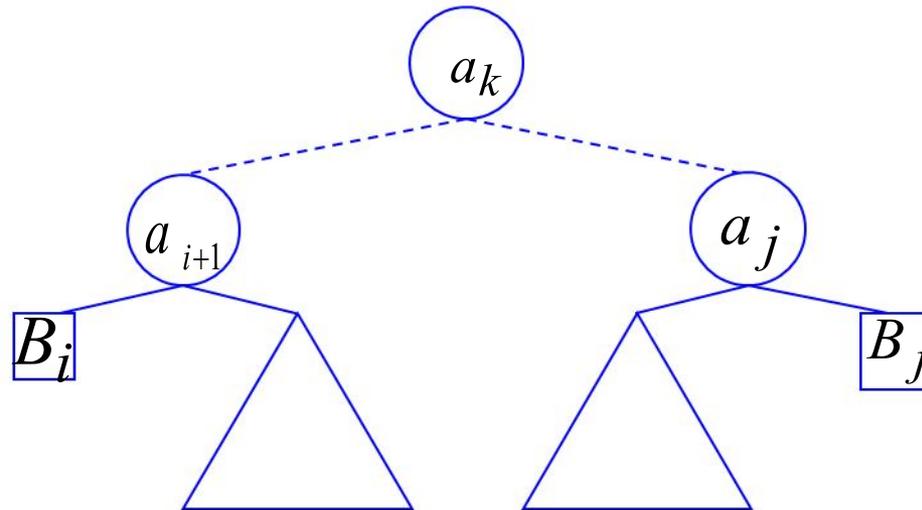


Рис. 2.10. Структура расширенного поддерева T_{ij}

$$C_{ij} = \min_{i < k \leq j} \{C_{i,k-1} + C_{k,j} + (w_{i,k-1} + w_{k,j} + p_k)\}$$

Преобразование

$$C_{ij} = \min_{i < k \leq j} \{C_{i,k-1} + C_{k,j} + (w_{i,k-1} + w_{k,j} + p_k)\}$$

$$w_{i,k-1} + w_{k,j} + p_k = w_{ij}$$

+

w_{ij} не зависит от структуры поддерева T_{ij}

$$C_{ij} = \min_{i < k \leq j} \{C_{i,k-1} + C_{k,j}\} + w_{ij}$$

1) $C_{ij} = 0$

2) разности индексов $k-1-i$ и $j-k$ в слагаемых $C_{i,k-1}$ и C_{j-k} меньше, чем разность индексов $j-i$ в C_{ij} .

3) $L = j - i, L = 0..n$

Таблица

(аналогия с задачей о перемножении цепочки матриц)

$l = 0$	$T(0, 0)$	$T(1, 1)$	$T(2, 2)$	$T(3, 3)$...	$T(n-1, n-1)$	$T(n, n)$
$l = 1$	$T(0, 1)$	$T(1, 2)$	$T(2, 3)$...	$T(n-2, n-1)$	$T(n-1, n)$	
$l = 2$	$T(0, 2)$	$T(1, 3)$...	$T(n-3, n-1)$	$T(n-2, n)$		
...			
$l = n-2$	$T(0, n-2)$	$T(1, n-1)$	$T(2, n)$				
$l = n-1$	$T(0, n-1)$	$T(1, n)$					
$l = n$	$T(0, n)$						

$T_{ij}:$

$w_{ij} =$
 $C_{ij} =$
 $Num_{ij} =$

$$\arg \min_{i < k \leq j} \{C_{i,k-1} + C_{k,j}\} = Num$$

```

for (i = 0; i < n; i++) {c[i][i] = 0; w[i][i] = q[i];} //заполнена первая строка
for (L=1; L<n; L++) {
    for (i=0; i<n-L; i++) {
        j = i + L; // заполнение T(i, j):
        w[i][j] = w[i][j - 1] + (p[j]+q[j]);
        c[i][j] = +∞;
        for (k = i + 1 ; k < j - 1; k++) {
s = c[i][k - 1] + c [k][j];
            if (s < c[i][j]) {
                c[i][j] = s;
                num[i][j] = k
            };
        }
        c[i][j] = c[i][j] + w[i][j];
    }
}
}

```

Вычисление
таблицы

$n^2/2$ элементов
памяти и $n^3/3$
выполнений тела
внутреннего цикла

См. пример в файле «2_08_ОДП.doc»
С.67,68-...

Построение БДП по таблице значений num

```
BinT MakeOptBST (int i, j )
{  int k; ElemBinT root;
   BinT L, R;
   k = num[i ][j ]; root = a[k];
   if (i < k -1) L = MakeOptBST (i, k -1);
   else L = NULL;
   if (k < j ) R = MakeOptBST (k, j );
   else R = NULL;
   return ConsBT (root, L, R);
} //MakeOptBST
```

со стартовым вызовом $T = \text{MakeOptBST}(0, n)$.

Модификация Д.Кнута

$$r_{i,j-1} \leq r_{ij} \leq r_{i+1,j}$$

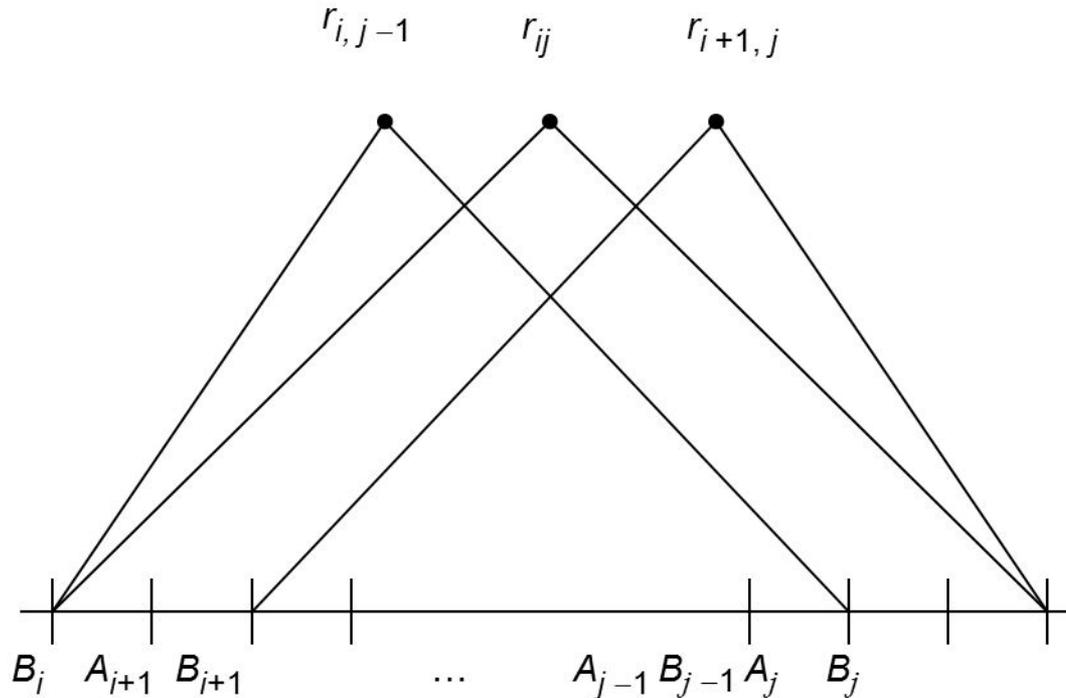


Рис. 2.14. Соотношение между корнями поддеревьев оптимального БДП

Вместо $k = (i+1) .. j \Rightarrow k = \text{num}[i][j-1] .. \text{num}[i+1][j]$

Так в ранее рассмотренном примере на последнем шаге при вычислении $C_{0,4}$ вместо рассмотрения четырёх кандидатов на роль корня дерева (см. с.70)

$$a_k \quad (k = 1, 2, 3, 4)$$

можно ограничиться лишь двумя (a_1 и a_2), поскольку

$$\text{num}[0, 3] \leq k \leq \text{num}[1, 4],$$

$$\text{а } \text{num}[0, 3] = 1 \text{ и } \text{num}[1, 4] = 2.$$

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ