

## Практическое занятие 1 Оптимальные деревья двоичного поиска

Пусть в построении дерева участвуют  $n$  различных упорядоченных записей  $k_1 < k_2 < \dots < k_n$ ,

Поставим задачу при фиксированных  $k_i$ ,  $\beta_i$  и  $\alpha_i$  построить двоичное дерево поиска, в котором среднее число сравнений при поиске будет минимальным. Обозначим узлы дерева, которые не хранят значений и соответствующие неудачному поиску (листья), через  $y_0, y_1, y_2, \dots, y_n$ . Тогда среднее число сравнений определяется следующим образом:

$$\sum_{i=1}^n \beta_i (\text{уровень}(k_i) + 1) + \sum_{i=0}^n \alpha_i \text{уровень}(y_i).$$

**Определение 1.** Назовем среднее число сравнений *ценой дерева*, а дерево с минимальной ценой – *оптимальным*.

**Определение 2.** Пусть *взвешенная длина пути* в  $T$

$$|T| = \sum_{i=1}^n \beta_i \cdot \text{уровень}(k_i) + \sum_{i=0}^n \alpha_i \cdot \text{уровень}(y_i).$$

Таким образом, цена дерева  $T$  есть

$$|T| + \sum_{i=1}^n \beta_i$$

и поскольку  $\sum_{i=1}^n \beta_i$  не зависит от структуры  $T$ , можно сосредоточить свое внимание на величине  $|T|$ .

Идея оптимальных деревьев была впервые развита Э.Н. Гильбертом и Э.Ф. Муром [10, 19, 40, 43], которые предложили метод их построения.

Отыщем оптимальное двоичное дерево поиска над множеством ключей с данными частотами. При данных весах узлов  $\beta_i \geq 0$  ( $1 \leq i \leq n$ ) и весах листьев  $\alpha_i \geq 0$ , ( $0 \leq i \leq n$ ) необходимо построить двоичное дерево поиска  $T$  над упорядоченным множеством всех ключей  $k_1 < k_2 < \dots < k_n$ , такое, что взвешенная длина пути дерева минимальна.

Эта задача дает яркий пример эффективности динамического программирования. Поиск оптимального дерева можно было бы осуществить перебором всех возможных деревьев из  $n$  элементов, однако их количество примерно равно  $\frac{4^n}{n\sqrt{\pi n}}$ . Метод динамического программирования

позволяет построить полиномиальный алгоритм, используя следующие факты:

1. Оптимальное двоичное дерево поиска  $T$  с весами  $\alpha_0, \beta_1, \alpha_1, \beta_2, \dots, \beta_n, \alpha_n$  имеет некоторый вес  $\beta_i$  в корне, левым поддеревом корня является оптимальное двоичное дерево поиска  $T_l$  с весами  $\alpha_0, \beta_1, \alpha_1, \beta_2, \dots, \beta_{i-1}, \alpha_{i-1}$  и правым поддеревом корня является оптимальное двоичное дерево поиска  $T_r$  с весами  $\alpha_i, \beta_{i+1}, \dots, \beta_n, \alpha_n$ .
2. Цену дерева  $T$  можно вычислить с помощью информации о двух поддеревьях  $T_l$  и  $T_r$ . В частности,  $|T|$  можно вычислить по взвешенным длинам путей  $|T_l|$  и  $|T_r|$  и суммам всех их весов  $W_l$  и  $W_r$  соответственно:

$$|T| = |T_l| + |T_r| + W_l + W_r,$$

где  $W_l = \sum_{j=1}^{i-1} \beta_j + \sum_{j=0}^{i-1} \alpha_j$  и  $W_r = \sum_{j=i+1}^n \beta_j + \sum_{j=i}^n \alpha_j$  – суммы весов элементов, входящих в левое и

правое деревья, соответственно.

Принцип оптимальности гласит, что для того, чтобы выбрать корень оптимального дерева, нужно вычислить для каждого узла  $k_i$  с весом  $\beta_i$  взвешенную длину пути в предположении, что  $k_i$  является корнем  $T$ . Это требует знания оптимальных правого и левого поддеревьев  $k_i$ . Если эти вычисления проводятся по рекурсивной схеме сверху вниз, то одни и те же оптимальные деревья

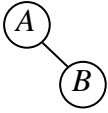
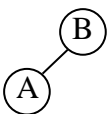
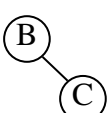
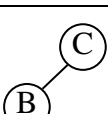
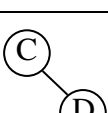
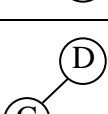
будут вычисляться повторно. Следовательно, алгоритм построения оптимального дерева лучше всего организовать снизу вверх. При этом для  $1 \leq i \leq n$  строится  $n + i - 1$  оптимальное дерево на  $i$  последовательных узлах (и смежных с ним листьях), используя построенные перед этим оптимальные деревья на  $i - 1$  или меньшем количестве последовательных узлов. Построение осуществляется перебором всех возможных вариантов корней. Пересчитываются взвешенные длины путей, и выбирается дерево с минимальной взвешенной длиной пути. Таким образом, каждое оптимальное поддерево на последовательных весах строится ровно один раз. Начало построения при  $i = 1$  тривиально, поскольку на одном узле существует только одно дерево. Когда процесс заканчивается при  $i = n$ , получено искомое оптимальное дерево. Поясним, почему поддерева, содержащие не последовательно идущие узлы, не рассматриваются. Пусть поддерево содержит узлы  $k_i < k_{i+1} < \dots < k_{j-1} < k_{j+1} < \dots < k_m$  и не содержит  $k_j$ . Тогда все элементы данного поддерева больше или меньше  $k_j$  одновременно, а это невозможно, так как  $k_1 < k_2 < \dots < k_n$ .

Подробно рассмотрим работу алгоритма при построении оптимального дерева на четырех ключах  $A, B, C, D$  с весами узлов 6, 2, 8, 7, соответственно, и весами листьев, равными нулю.

Заметим, что пустые поддерева и поддерева, содержащие один узел, являются оптимальными.

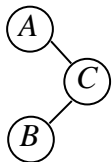
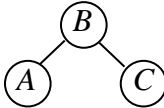
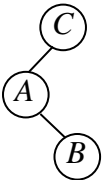
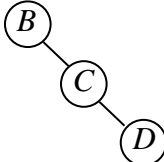
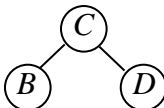
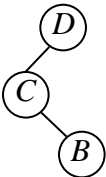
На первом этапе строятся все оптимальные деревья, содержащие два узла. Рассматриваются попарно все подряд стоящие узлы и строятся поддерева перебором возможных корней. Подсчитываются взвешенные длины путей, и выбирается дерево с минимальной взвешенной длиной пути для каждой пары узлов. Построение оптимальных поддеревьев на двух узлах приведено в табл. 1.1.

Табл. 1.1 Построение оптимальных поддеревьев на двух узлах

Узлы	Корень	Вид дерева	Взвешенная длина пути	Примечание
A,B	A		2	оптимально
	B		6	
B,C	B		8	
	C		2	оптимально
C,D	C		7	оптимально
	D		8	

На втором этапе строятся оптимальные поддерева, содержащие три узла. Рассматриваются теперь уже три подряд стоящих узла, и строятся оптимальные поддерева перебором возможных корней и с учётом уже имеющихся оптимальных поддеревьев размерами 1 и 2, поскольку поддерева оптимального дерева должны быть оптимальными (табл. 1.2).

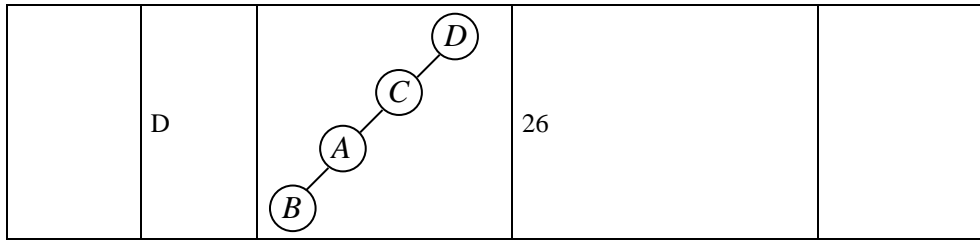
Табл. 1.2 Построение оптимальных поддеревьев на трёх узлах

Узлы	Корень	Вид дерева	Взвешенная длина пути	Примечание
A,B,C	A		12	
	B		14	
	C		10	оптимально
B,C,D	B		22	
	C		9	оптимально
	D		12	

На итоговом 3 этапе в нашем примере строится искомое оптимальное дерево из 4-х узлов. Аналогично предыдущему шагу перебираются все возможные корни и используются все имеющиеся оптимальные деревья размерами 1, 2 и 3.

Табл. 1.3 Построение оптимальных поддеревьев на четырёх узлах

Узлы	Корень	Вид дерева	Взвешенная длина пути	Примечание
A,B,C, D	A		26	
	B		28	
	C		17	оптимально



Приведённый пример иллюстрирует работу алгоритма, показывая, что оптимальное дерево из 4-х элементов строится на основе построенных оптимальных поддеревьев для одного, двух и трех элементов. Хотя в этом примере имеется единственное оптимальное дерево, в общем случае единственность не имеет места.

Когда веса листьев  $\alpha_i$  не равны нулю, удобно рассматривать верхний треугольник в  $(n+1) \times (n+1)$ -матрице. Каждый элемент матрицы содержит три величины. Для  $0 \leq i \leq j \leq n$  имеем

$R_{ij}$  – номер корня оптимального дерева на последовательности весов  $\alpha_i, \beta_{i+1}, \alpha_{i+1}, \dots, \beta_j, \alpha_j$ ,

$W_{ij}$  – сумма весов  $\alpha_i + \beta_{i+1} + \dots + \beta_j + \alpha_j$ ,

$P_{ij}$  – взвешенная длина пути оптимального дерева на последовательности весов  $\alpha_i, \beta_{i+1}, \alpha_{i+1}, \dots, \beta_j, \alpha_j$  (когда  $i=j$ , дерево состоит из одного листа и эта величина равна нулю).

Заметим, что для вычисления элемента этой матрицы с номером  $i, j$  нам нужны элементы с номерами  $u=i, v \leq j$  и  $v=j, u \geq i$ . Поэтому структура алгоритма проста. Два внешних цикла просматривают все элементы над диагональю  $(n+1) \times (n+1)$ -матрицы. Присваивающее предложение

$R_{ij}$  – любое значение  $k, i < k \leq j$ , минимизирующее сумму  $P_{i,k-1} + P_{k,j} + W_{i,k-1} + W_{k,j}$

приводит к еще одному вложенному циклу, который в общем случае выполняется  $j-i$  раз. Число операций алгоритма равно поэтому  $\frac{1}{6}n^3 + O(n^2)$ . Таким образом, для алгоритма построения дерева снизу вверх основанного на принципе оптимальности, требуется  $O(n^3)$  операций и память размера  $O(n^2)$ .

Алгоритм использует немногие свойства оптимальных деревьев бинарного поиска: фактически только специальный вид выражений в правой части присваивающего предложения зависит от свойств деревьев.

### 1.1 Алгоритм построения оптимального дерева поиска снизу вверх при данных частотах успешного и безуспешного поиска:

[установить начальные данные (заполнить главную диагональ матрицы)]

for  $i=0$  to  $n$  do {

$R_{ii} := i$ ;

$W_{ii} := \alpha_i$ ;

$P_{ii} := 0$ ;

}

[посетить каждую побочную диагональ, выше главной]

for  $l=1$  to  $n$  do

[посетить каждый элемент на побочной диагонали]

for  $i=0$  to  $n-l$  do

{  $j := i+l$ ;

$R_{ij} :=$  любое значение  $k, i < k \leq j$ ,

минимизирующее сумму  $P_{i,k-1} + P_{k,j} + W_{i,k-1} + W_{k,j}$

$W_{ij} := W_{i,j-1} + \beta_j + \alpha_j$

$P_{ij} := \min_{i < k \leq j} (P_{i,k-1} + P_{k,j} + W_{i,k-1} + W_{k,j})$

}

Анализируя более тщательно специфические свойства оптимальных деревьев бинарного поиска, можно улучшить алгоритм так, чтобы он выполнялся за время  $O(n^2)$  вместо  $O(n^3)$ . Ключом к такому улучшению является теорема:

**Теорема.** Пусть  $R_{i,j-1}$  – корень оптимального дерева над  $\alpha_i, \beta_{i+1}, \dots, \beta_{j-1}, \alpha_{j-1}$  и  $R_{i+1,j}$  – корень оптимального дерева над  $\alpha_{i+1}, \beta_{i+2}, \dots, \beta_j, \alpha_j$ , где  $i < j-1$ . Тогда над  $\alpha_i, \beta_{i+1}, \dots, \beta_j, \alpha_j$  существует оптимальное дерево, корень которого  $R_{ij}$  удовлетворяет неравенствам

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

**Без доказательства.**

Согласно этой теореме, самый внутренний цикл (по  $k$ ) выполняется *nej-iraz*, а меньше.

Теорема позволяет уточнить предложение алгоритма так:

$R_{i,j} :=$  любое значение  $k$ ,  $R_{i,j-1} \leq k \leq R_{i+1,j}$ , минимизирующее сумму

$$P_{i,k-1} + P_{ij} + W_{i,k-1} + W_{kj}$$

чтобы получить алгоритм построения оптимальных деревьев бинарного поиска с временем работы  $O(n^2)$ .

**Задача:** Построить оптимальное двоичное дерево поиска на ключах A(1), B(2), C(3), D(4) с весами узлов (успешный поиск)  $\beta_1=5, \beta_2=4, \beta_3=1, \beta_4=7$  и листьев (безуспешный поиск)  $\alpha_0=3, \alpha_1=8, \alpha_2=2, \alpha_3=9, \alpha_4=6$ .

**Решение:** Воспользуемся алгоритмом, построим матрицу  $5 \times 5$  ( $n+1=5$ )

Заполняем главную диагональ матрицы:

	0	1	2	3	4
0	$R_{00}=1$ $W_{00}=\alpha_0=3$ $P_{00}=0$				
1		$R_{11}=1$ $W_{11}=\alpha_1=8$ $P_{11}=0$			
2			$R_{22}=2$ $W_{22}=\alpha_2=2$ $P_{22}=0$		
3				$R_{33}=3$ $W_{33}=\alpha_3=9$ $P_{33}=0$	
4					$R_{44}=4$ $W_{44}=\alpha_4=6$ $P_{44}=0$

Заполняем диагонали выше главной диагонали.

$$l=1; i=0,1,2,3; j=i+l$$

Элемент  $M_{01}$  :

$$0 < k \leq 1 \rightarrow k=1$$

$$R_{01}=1; W_{01}=W_{00}+\beta_1+\alpha_1 = 3+5+8=16; P_{01}=P_{00}+P_{11}+W_{00}+W_{11}=0+0+3+8=11$$

Элемент  $M_{12}$  :

$$1 < k \leq 2 \rightarrow k=2$$

$$R_{12}=2; W_{12}=W_{11}+\beta_2+\alpha_2 = 8+4+2=14; P_{12}=P_{11}+P_{22}+W_{11}+W_{22}=0+0+8+2=10$$

Элемент  $M_{23}$  :

$$2 < k \leq 3 \rightarrow k=3$$

$$R_{23}=3; W_{23}=W_{22}+\beta_3+\alpha_3 = 2+1+9=12; P_{23}=P_{22}+P_{33}+W_{22}+W_{33}=0+0+2+9=11$$

Элемент  $M_{34}$  :

$$3 < k \leq 4 \rightarrow k=4$$

$$R_{34}=4; W_{34}=W_{33}+\beta_4+\alpha_4 = 9+7+6=22; P_{34}=P_{33}+P_{44}+W_{33}+W_{44}=0+0+9+6=15$$

	0	1	2	3	4
0	$R_{00}=1$ $W_{00}=\alpha_0=3$ $P_{00}=0$	$R_{01}=1$ $W_{01}=16$ $P_{01}=11$			
1		$R_{11}=1$ $W_{11}=\alpha_1=8$ $P_{11}=0$	$R_{12}=2$ $W_{12}=14$ $P_{12}=10$		

2			$R_{22}=2$ $W_{22}=\alpha_2=2$ $P_{22}=0$	$R_{23}=3$ $W_{23}=12$ $P_{23}=11$	
3				$R_{33}=3$ $W_{33}=\alpha_3=9$ $P_{33}=0$	$R_{34}=4$ $W_{34}=22$ $P_{34}=15$
4					$R_{44}=4$ $W_{44}=\alpha_4=6$ $P_{44}=0$

$l=2; i=0,1,2; j=i+l$

Элемент  $M_{02}$  :

$$W_{02}=W_{01}+\beta_2+\alpha_2 = 16+4+2=22;$$

$$0 < k \leq 2 \rightarrow k=1 \text{ или } k=2$$

Если  $k=1$

$$P_{02}=P_{00}+P_{12}+W_{00}+W_{12}=0+10+3+14=27$$

Если  $k=2$

$$P_{02}=P_{01}+P_{22}+W_{01}+W_{22}=11+0+16+2=29$$

Т.к.  $27 < 29$ , то берем  $k=1$ ,  $R_{02}=1$

Элемент  $M_{13}$  :

$$W_{13}=W_{12}+\beta_3+\alpha_3 = 14+1+9=24;$$

$$1 < k \leq 3 \rightarrow k=2 \text{ или } k=3$$

Если  $k=2$

$$P_{13}=P_{11}+P_{23}+W_{11}+W_{23}=0+11+8+12=31$$

Если  $k=3$

$$P_{13}=P_{12}+P_{33}+W_{12}+W_{33}=10+0+14+9=33$$

Т.к.  $31 < 33$ , то берем  $k=2$ ,  $R_{13}=2$

Элемент  $M_{24}$  :

$$W_{24}=W_{23}+\beta_4+\alpha_4 = 12+7+6=25;$$

$$2 < k \leq 4 \rightarrow k=3 \text{ или } k=4$$

Если  $k=3$

$$P_{24}=P_{22}+P_{34}+W_{22}+W_{34}=0+15+2+22=39$$

Если  $k=4$

$$P_{24}=P_{23}+P_{44}+W_{23}+W_{44}=11+0+12+6=29$$

Т.к.  $29 < 39$ , то берем  $k=4$ ,  $R_{24}=4$

	0	1	2	3	4
0	$R_{00}=0$ $W_{00}=\alpha_0=3$ $P_{00}=0$	$R_{01}=1$ $W_{01}=16$ $P_{01}=11$	$R_{02}=1$ $W_{02}=22$ $P_{02}=27$		
1		$R_{11}=1$ $W_{11}=\alpha_1=8$ $P_{11}=0$	$R_{12}=2$ $W_{12}=14$ $P_{12}=10$	$R_{13}=2$ $W_{13}=24$ $P_{13}=31$	
2			$R_{22}=2$ $W_{22}=\alpha_2=2$ $P_{22}=0$	$R_{23}=3$ $W_{23}=12$ $P_{23}=11$	$R_{24}=4$ $W_{24}=25$ $P_{24}=29$
3				$R_{33}=3$ $W_{33}=\alpha_3=9$ $P_{33}=0$	$R_{34}=4$ $W_{34}=22$ $P_{34}=15$
4					$R_{44}=4$ $W_{44}=\alpha_4=6$ $P_{44}=0$

$l=3; i=0,1; j=i+l$

Элемент  $M_{03}$  :

$$W_{03}=W_{02}+\beta_3+\alpha_3 = 22+1+9=32;$$

$0 < k \leq 3 \rightarrow k=1$  или  $k=2$  или  $k=3$

Если  $k=1$

$$P_{03}=P_{00}+P_{13}+W_{00}+W_{13}=0+31+3+24=58$$

Если  $k=2$

$$P_{03}=P_{01}+P_{23}+W_{01}+W_{23}=11+11+16+12=50$$

Если  $k=3$

$$P_{03}=P_{02}+P_{33}+W_{02}+W_{33}=27+0+22+9=58$$

Т.к.  $\min(58,50,58)=50$ , то берем  $k=2$ ,  $R_{03}=2$

Элемент  $M_{14}$  :

$$W_{14}=W_{13}+\beta_4+\alpha_4 = 24+7+6=37;$$

$1 < k \leq 4 \rightarrow k=2$  или  $k=3$  или  $k=4$

Если  $k=2$

$$P_{14}=P_{11}+P_{24}+W_{11}+W_{24}=0+29+8+25=62$$

Если  $k=3$

$$P_{14}=P_{12}+P_{34}+W_{12}+W_{34}=10+15+14+22=61$$

Если  $k=4$

$$P_{14}=P_{13}+P_{44}+W_{13}+W_{44}=31+0+24+6=61$$

Т.к.  $\min(62,61,61)=61$ , то берем  $k=4$ ,  $R_{14}=4$

$l=4$ ;  $i=0$ ;  $j=i+l$

Элемент  $M_{04}$  :

$$W_{04}=W_{03}+\beta_4+\alpha_4 = 32+7+6=45;$$

$0 < k \leq 4 \rightarrow k=1$  или  $k=2$  или  $k=3$  или  $k=4$

Если  $k=1$

$$P_{04}=P_{00}+P_{14}+W_{00}+W_{14}=0+61+3+37=101$$

Если  $k=2$

$$P_{04}=P_{01}+P_{24}+W_{01}+W_{24}=11+29+16+25=81$$

Если  $k=3$

$$P_{04}=P_{02}+P_{34}+W_{02}+W_{34}=27+15+22+22=86$$

Если  $k=4$

$$P_{04}=P_{03}+P_{44}+W_{03}+W_{44}=50+0+32+6=88$$

Т.к.  $\min(101,81,86,88)=81$ , то берем  $k=2$ ,  $R_{04}=2$

	0	1	2	3	4
0	$R_{00}=0$ $W_{00}=\alpha_0=3$ $P_{00}=0$	$R_{01}=1$ $W_{01}=16$ $P_{01}=11$	$R_{02}=1$ $W_{02}=22$ $P_{02}=27$	$R_{03}=2$ $W_{03}=32$ $P_{03}=50$	$R_{04}=2$ $W_{04}=45$ $P_{04}=81$
1		$R_{11}=1$ $W_{11}=\alpha_1=8$ $P_{11}=0$	$R_{12}=2$ $W_{12}=14$ $P_{12}=10$	$R_{13}=2$ $W_{13}=24$ $P_{13}=31$	$R_{14}=4$ $W_{14}=37$ $P_{14}=61$
2			$R_{22}=2$ $W_{22}=\alpha_2=2$ $P_{22}=0$	$R_{23}=3$ $W_{23}=12$ $P_{23}=11$	$R_{24}=4$ $W_{24}=25$ $P_{24}=29$
3				$R_{33}=3$ $W_{33}=\alpha_3=9$ $P_{33}=0$	$R_{34}=4$ $W_{34}=22$ $P_{34}=15$
4					$R_{44}=4$ $W_{44}=\alpha_4=6$ $P_{44}=0$

Используя данную матрицу, построим оптимальное дерево поиска:

Так как  $R_{04}=2$ , то корнем оптимального дерева будет вершина  $B(2)$ , вставляем этот узел в двоичное дерево поиска. Оптимальным левым поддеревом будет поддерево с корнем в  $A(1)$  (т.к.  $R_{01}=1$ ), вставляем этот узел в двоичное дерево поиска. А оптимальным правым поддеревом будет

поддерево с корнем в  $D(4)$  (т.к.  $R_{34}=4$ ), вставляем этот узел в двоичное дерево поиска. У  $D$  правого поддерева нет, а оптимальное левое поддерево имеет корень  $C$ , вставляем этот узел в дерево. Получили оптимальное двоичное дерево поиска:

$B(2) \quad \beta_2=4$

$\beta_1=5 \quad A(1)$

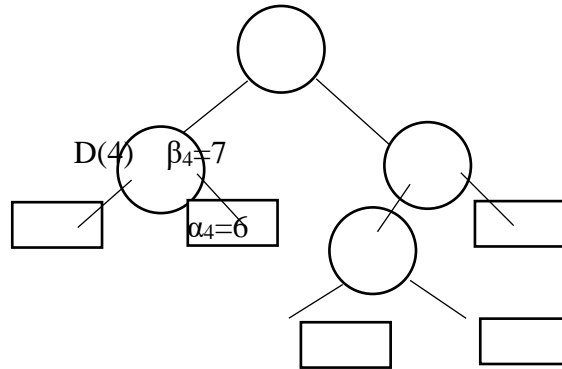
$\alpha_0=3$

$C(3) \quad \beta_3=1$

$\alpha_2=2$

$\alpha_1=8$

$\alpha_3=9$



**Задание:** Реализовать на языке C++ алгоритм построения оптимального дерева поиска снизу вверх при данных частотах успешного и безуспешного поиска.