Глава 2. Деревья
п3. Оптимальное дерево поиска

Оптимальное дерево поиска

Дано:

n – количество ключей,

$\mathbf{d_1} < \mathbf{d_2} < \ldots < \mathbf{d_n}$ – упорядоченное множество ключей

$\mathbf{p_1}$, $\mathbf{p_2}$, …, $\mathbf{p_n}$ – частоты (вероятности), с которыми эти ключи появляются на входе процедуры поиска.

Например:

$n = 3$,

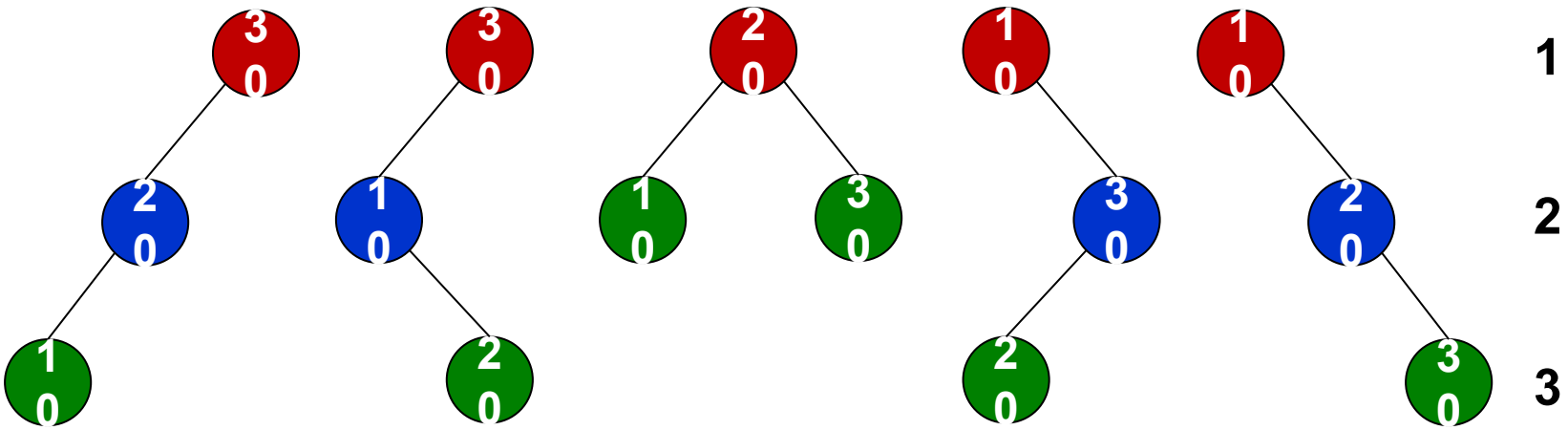
$d_1 = 10$, $d_2 = 20$, $d_3 = 30$

$p_1 = 5$, $p_2 = 3$, $p_3 = 2$ ($p_1 = 0.5$, $p_2 = 0.3$, $p_3 = 0.2$)

Требуется построить дерево поиска для заданного множества ключей такое, чтобы ключи наиболее часто запрашиваемые находились как можно быстрее.

Оптимальное дерево поиска

Один из методов построения – полный перебор:



5*3 +	5*2 +	5*2 +	5*1 +	5*1 +
3*2 +	3*3 +	3*1 +	3*3 +	3*2 +
2*1 =	2*1 =	2*2 =	2*2 =	2*3 =
= 23	= 21	= 17	= 18	= 17

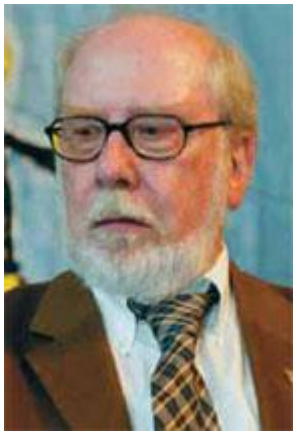
n = 3,

$d_1 = 10$, $d_2 = 20$, $d_3 = 30$

$p_1 = 5$, $p_2 = 3$, $p_3 = 2$ ($p_1 = 0.5$, $p_2 = 0.3$, $p_3 = 0.2$)

Оптимальное дерево поиска

Никлаус Вирт:



«Учитывая, что число возможных конфигураций из n вершин растет экспоненциально с ростом n, задача построения оптимального дерева при больших n кажется совершенно безнадежной.»

Оптимальное дерево поиска

Никлаус Вирт:

«Однако оптимальные деревья обладают одним важным свойством, которое помогает их обнаруживать:
Все их поддеревья тоже оптимальны!»

Оптимальное дерево поиска

Это позволяет разработать алгоритм, который систематически находит всё бóльшие и бóльшие поддеревья, начиная с отдельных вершин, как наименьших возможных поддеревьев.

Таким образом, дерево растет
«от листьев к корню», «снизу вверх»…
Н. Вирт. Алгоритмы и структуры данных.

Примеры деревьев поиска

Дерево поиска, используемое в трансляторах для опознания служебных слов (есть часто используемые – int, for, есть реже – enum, union).

База данных библиотеки (есть часто запрашиваемые книги, есть поднимаемые раз в 10 лет, есть часто запрашиваемые книги, которых в библиотеке нет).

В оптимальном дереве поиска учитываются частоты поиска различных ключей:

1) как входящих в дерево,

2) так и тех, которых нет в дереве.

Оптимальное дерево поиска

Дано:

n – количество ключей,

$d_1 < d_2 < \ldots < d_n$ – упорядоченное множество ключей

$p_1, p_2, \ldots, p_n$ – частоты (вероятности), с которыми эти ключи появляются на входе процедуры поиска (удобнее использовать частоты).

Также даны «промежуточные» частоты:

q_i – частота поиска значений между d_i и d_{i+1} ($d_i < x < d_{i+1}$).

q_0 – частота поиска значений меньше d_0 ($x < d_0$).

q_n – частота поиска значений больше d_n ($x > d_n$).

Пример оптимального дерева поиска

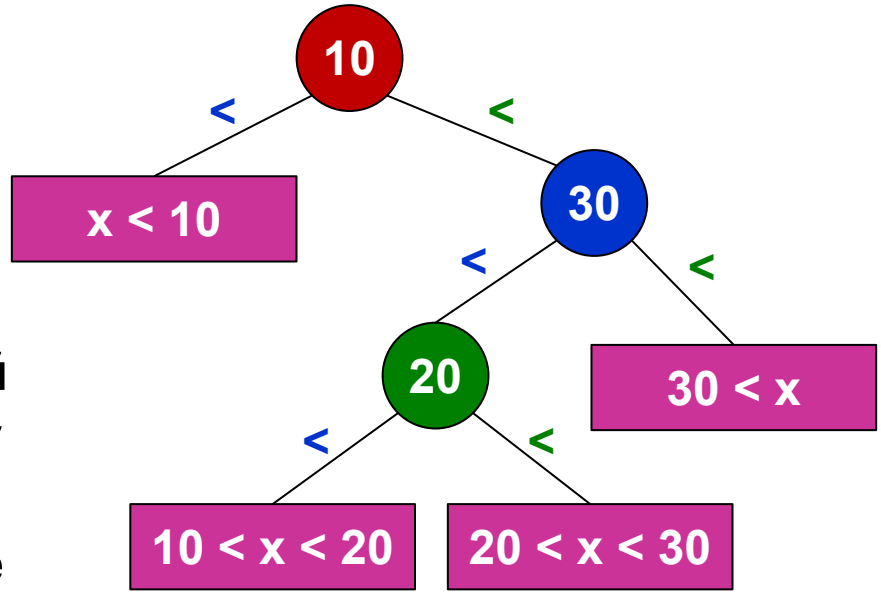
Пример исходных данных:

d: 10, 20, 30

p: 10, 3, 2

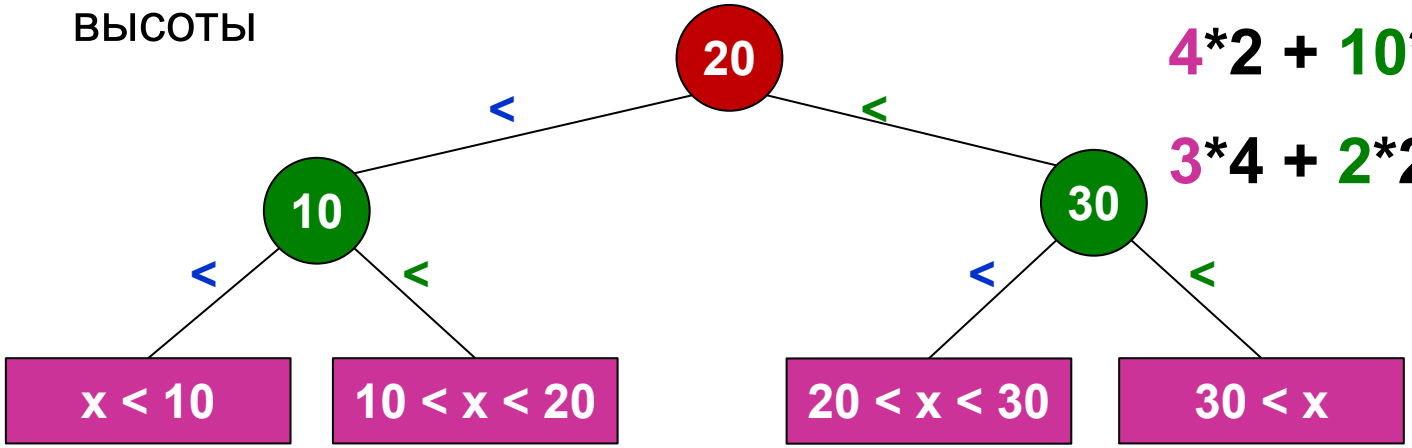
q: 4 2 3 4

Суммарная длина (цена) путей поиска = сумме частот (зелёные у узлов, пурпурные у ловушек), умноженных на соответствующие высоты



4*2 + 10*1 + 2*4 + 3*3 +

3*4 + 2*2 + 4*3 = 63



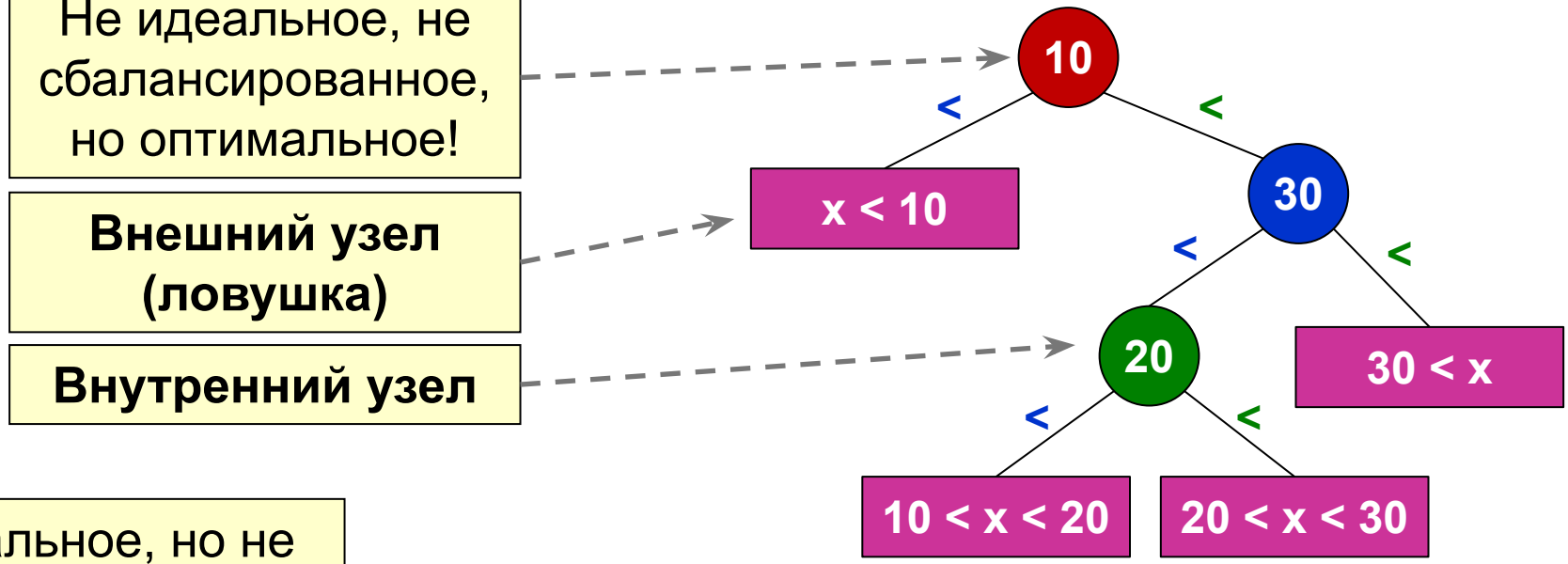
4*3 + 10*2 + 2*3 + 3*1 + 3*3 + 2*2 + 4*3 = 66

Пример оптимального дерева поиска

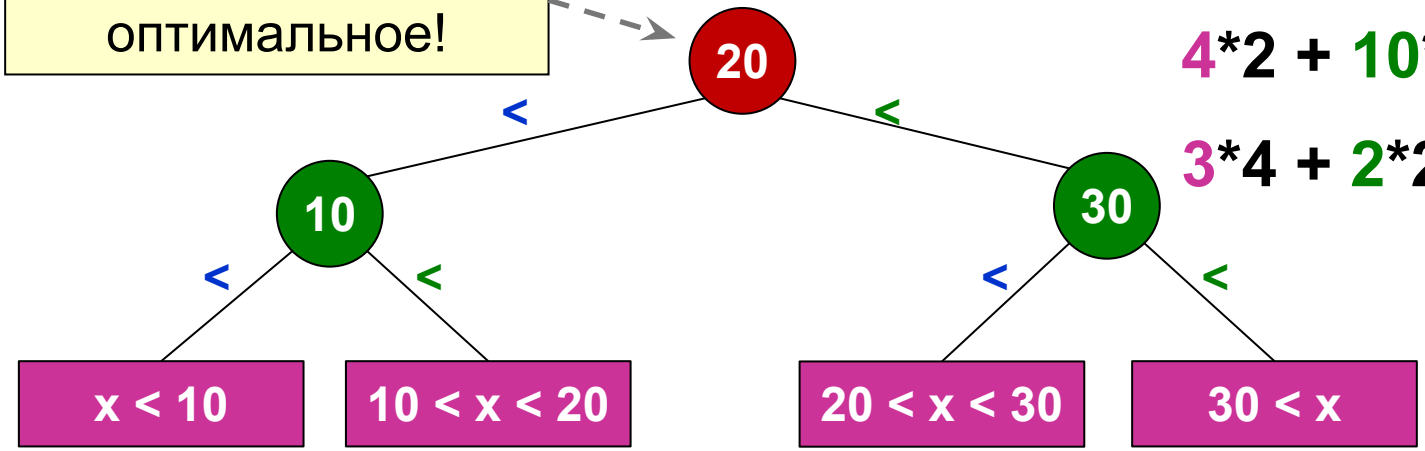
Не идеальное, не сбалансированное, но оптимальное!

Внешний узел (ловушка)

Внутренний узел



Идеальное, но не оптимальное!



4*2 + 10*1 + 2*4 + 3*3 + 3*4 + 2*2 + 4*3 = 63

4*3 + 10*2 + 2*3 + 3*1 + 3*3 + 2*2 + 4*3 = 66

Построение оптимального дерева поиска

Требуется: построить дерево поиска для заданного множества ключей, минимизирующее математическое ожидание числа сравнений при поиске, т.е:

$$C = \sum_{i=1}^{n} p_i \times h_i + \sum_{j=0}^{n} q_j \times h'_j \text{, где}$$

h_i - высота внутреннего узла **i** (узла дерева с ключом d_i);

h'_j - высота внешнего узла **j** («ловушки» для $d_j < x < d_{j+1}$).

Полученная величина C называется ***общей взвешенной длиной*** (***стоимостью***, ***ценой***) ***путей дерева***.

Дерево поиска является оптимальным, если его стоимость является минимальной.

Построение оптимального дерева поиска

Утверждение:

В оптимальном дереве все поддеревья также являются оптимальными.

Доказательство:

Рассмотрим оптимальное дерево **T**. Предположим, что его левое поддерево не является оптимальным.

Заменим левое поддерево на оптимальное, получив дерево T*. Вклад вершин левого поддерева в общую стоимость дерева T* меньше, чем у T. Следовательно, исходное дерево T не является оптимальным.

Аналогично для правого поддерева.

Утверждение доказано.

Построение оптимального дерева поиска

C(T) – стоимость оптимального дерева с корнем T. Её можно представить в виде суммы стоимостей левого (L) и правого (R) поддеревьев плюс стоимость фрагментов путей поиска, проходящих через корень:

$$C(T) = C(L) + C(R) + \left(\sum_{i=1}^{n} p_i + \sum_{j=0}^{n} q_j \right)$$

В дальнейшем сумму p_i и q_j будем называть **весом дерева**:

$$W = \sum_{i=1}^{n} p_i + \sum_{j=0}^{n} q_j$$

Построение оптимального дерева поиска

Введём следующие обозначения:

T_{ij} – оптимальное поддерево, состоящее из вершин с ключами $d_{i+1}, d_{i+2}, \dots, d_j$ и ловушек. Его стоимость – C_{ij}.

W_{ij} – стоимость фрагментов путей поиска, проходящих через корень поддерева T_{ij}, т.е. **вес поддерева T_{ij}**.

$$W = \sum_{i=1}^{n} p_i + \sum_{j=0}^{n} q_j \Rightarrow W_{ij} = \sum_{f=i+1}^{j} p_f + \sum_{h=i}^{j} q_h =$$

$$= \boldsymbol{q_i + p_{i+1} + q_{i+1} + \cdots + p_{j-1} + q_{j-1}} + p_j + q_j$$

$$\Rightarrow \boldsymbol{W_{ij} = W_{ij-1}} + p_j + q_j$$

T_{ii} – пустое поддерево, в которое попадают при поиске ключа $d_i < x < d_{i+1}$. Его вес $\boldsymbol{W_{ii} = q_i}$.

Построение оптимального дерева поиска

Тогда стоимость поддерева T_{ij} определяется через:

$$C_{ij} = W_{ij} + \min_{i<k\leq j}(C_{ik-1} + C_{kj}), 0 \leq i < j \leq n,$$
$$C_{ii} = q_i, 0 \leq i \leq n.$$

Обозначим через R_{ij} значение k, при котором достигается определённый выше минимум. При этом узел с ключом d_k будет являться корнем поддерева T_{ij}, C_{ik-1} – стоимость его левого поддерева, C_{kj} – правого.

Отметим, что стоимости этих поддеревьев будут вычислены раньше (дерево ведь строится «от листьев к корню»).

Алгоритм построения оптимального дерева поиска

- **1.** *Для деревьев, содержащих не более одной вершины:*

1.1. Для $\mathbf{0} \le \boldsymbol{i} \le \boldsymbol{n}$: $\boldsymbol{C_{ii} = W_{ii} = q_i}$

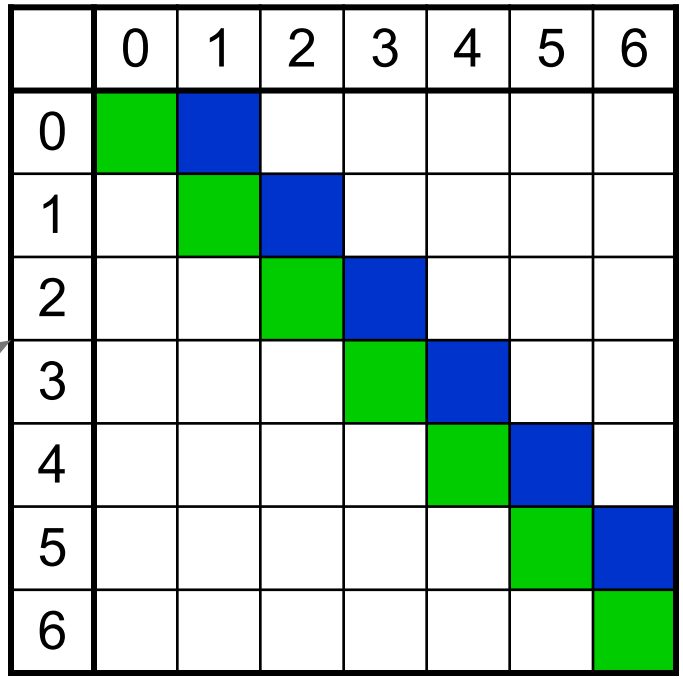
1.2. Для $\mathbf{0} \le \boldsymbol{i} \le \boldsymbol{n}$:

$\boldsymbol{W_{ij} = W_{ii} + p_j + q_j}$;

$\boldsymbol{C_{ij} = W_{ij} + C_{ii} + C_{jj}}$;

$\boldsymbol{R_{ij} = j}$.

Таких матриц будет три:
W – матрица весов
C – матрица стоимости
R – матрица номеров ключей в корнях поддеревьев



Алгоритм построения оптимального дерева поиска

- **2.** *Для деревьев, содержащих h вершин* $(2 \le h \le n)$:

2.1. Для $0 \le i \le n - h$:

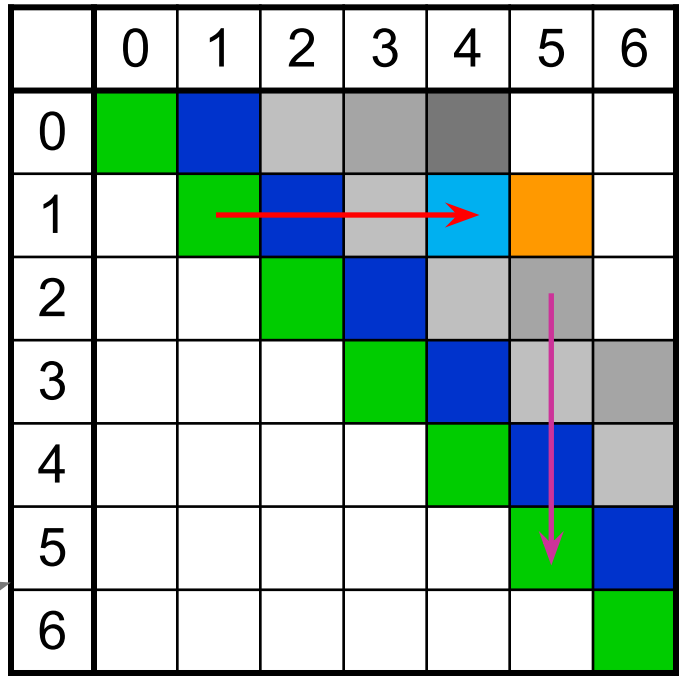
$j = i + h$;

$W_{ij} = W_{ij-1} + p_j + q_j$;

Найти $\min\limits_{i<k\le j}(C_{ik-1} + C_{kj})$;

$C_{ij} = W_{ij} + C_{ik-1} + C_{kj}$;

$R_{ij} = k$.



Порядок перебора пар для поиска k (h = 4, i = 1)

Обратите внимание, что к моменту вычисления значений в оранжевой клетке, значения предшествующих ей диагоналей уже посчитаны

Алгоритм построения оптимального дерева поиска

-

3. *Построить по таблице* ***R*** *дерево* T_{0n}:

Выполняется с помощью рекурсивной функции построения поддерева T_{ij}:

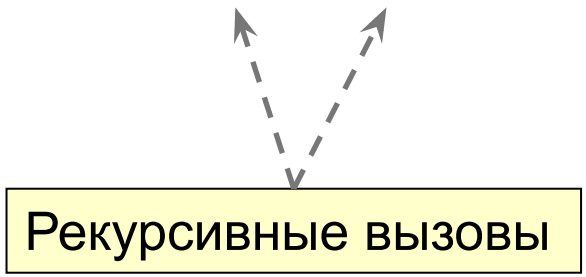
Начинаем с правого верхнего угла таблицы $(\mathbf{i = 0, j = n})$

3.1. Если $i \geq j$, то выйти и вернуть nullptr;

3.2. Иначе, $\mathbf{k = R_{ij}}$;

3.2. Создать узел с ключом $\mathbf{d_k}$ и потомками $\mathbf{T_{ik-1}}$, $\mathbf{T_{kj}}$;

3.3. Вернуть созданный узел.



Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1				
1		10			
2			1		
3				1	
4					10

C	0	1	2	3	4
0	1				
1		10			
2			1		
3				1	
4					10

R	0	1	2	3	4
0					
1					
2					
3					
4					

$$C_{ii} = W_{ii} = q_i, 0 \le i \le n$$

Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13			
1		10	12		
2			1	3	
3				1	16
4					10

C	0	1	2	3	4
0	1	24			
1		10	23		
2			1	5	
3				1	27
4					10

R	0	1	2	3	4
0		1			
1			2		
2				3	
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$C_{ij} = W_{ij} + C_{ii} + C_{jj}$$

$$R_{ij} = j$$

Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15		
1		10	12	14	
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39		
1		10	23	29	
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1		
1			2	2	
2				3	4
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{i<k\le j}(C_{ik-1} + C_{kj})$$

Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15	17	
1		10	12	14	29
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39	46	
1		10	23	29	68
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1	2	
1			2	2	4
2				3	4
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{i<k\le j}(C_{ik-1} + C_{kj})$$

Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15	17	32
1		10	12	14	29
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39	46	88
1		10	23	29	68
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1	2	4
1			2	2	4
2				3	4
3					4
4					

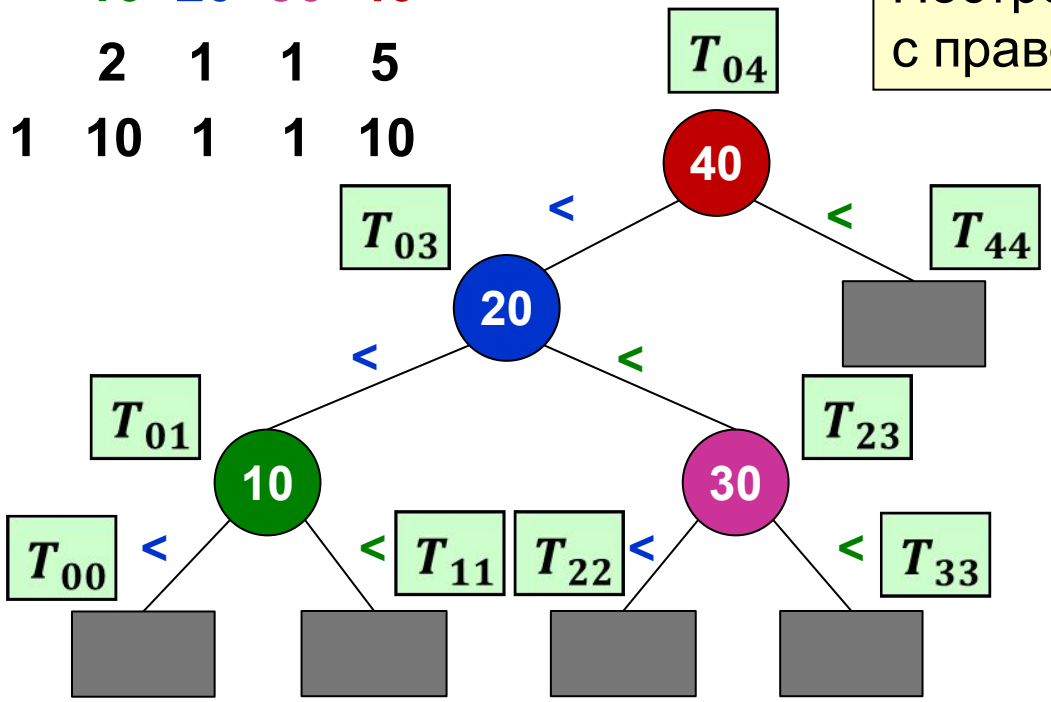
$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{i<k\le j}(C_{ik-1} + C_{kj})$$

Пример построения оптимального дерева поиска

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10



Построение дерева начинается с правого верхнего угла

R	0	1	2	3	4
0		1	1	2	4
1			2	2	4
2				3	4
3					4
4					

3.1. Если $\mathbf{i} \geq \mathbf{j}$, то выйти и вернуть nullptr;

3.2. Иначе, $\boldsymbol{k} = \boldsymbol{R_{ij}}$;

3.2. Создать узел с ключом $\boldsymbol{d_k}$ и потомками $\boldsymbol{T_{ik-1}}$, $\boldsymbol{T_{kj}}$;

3.3. Вернуть созданный узел.

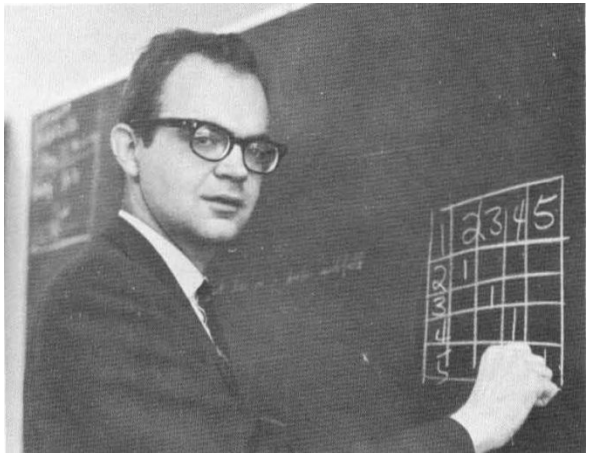
Алгоритмическая сложность построения оптимального дерева поиска

Для построения оптимального дерева поиска необходимо построить три таблицы размера $(n+1) \times (n+1)$. Для вычисления каждого элемента требуется $O(n)$ операций. Таким образом, общая сложность алгоритма - $O(n^3)$.

Однако сложность можно уменьшить до $O(n^2)$, изменив граничные условия для поиска минимума:

$$i + 1 \leq k \leq j \quad \rightarrow \quad R_{ij-1} \leq k \leq R_{i+1j}$$

Доказательство изложено в:
Knuth, Donald E. (1971)
"Optimum binary search trees",
Acta Informatica 1 (1): 14–25,
doi:10.1007/BF00264289



Пример с оптимизацией поиска C_{ij}

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15		
1		10	12	14	
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39		
1		10	23	29	
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1		
1			2	2	
2				3	4
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{R_{ij-1} \le k \le R_{i+1j}} (C_{ik-1} + C_{kj})$$

Пример с оптимизацией поиска C_{ij}

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15	17	
1		10	12	14	29
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39	46	
1		10	23	29	68
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1	2	
1			2	2	4
2				3	4
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{R_{ij-1} \le k \le R_{i+1j}} (C_{ik-1} + C_{kj})$$

Пример с оптимизацией поиска C_{ij}

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	1	10	1	1	10

индексы

Порядок вычислений **всегда** соответствует движению по диагонали сверху вниз ↘

W	0	1	2	3	4
0	1	13	15	17	32
1		10	12	14	29
2			1	3	18
3				1	16
4					10

C	0	1	2	3	4
0	1	24	39	46	88
1		10	23	29	68
2			1	5	33
3				1	27
4					10

R	0	1	2	3	4
0		1	1	2	4
1			2	2	4
2				3	4
3					4
4					

$$W_{ij} = W_{ij-1} + p_j + q_j$$

$$R_{ij} = k$$

$$C_{ij} = W_{ij} + \min_{R_{ij-1} \le k \le R_{i+1j}} (C_{ik-1} + C_{kj})$$

Примеры для самостоятельной работы

	0	1	2	3	4
d:		10	20	30	40
p:		2	1	1	5
q:	0	0	0	0	0

	0	1	2	3	4
d:		10	20	30	40
p:		0	0	0	0
q:	1	10	1	1	10

	0	1	2	3	4
d:		10	20	30	40
p:		2	0	1	0
q:	1	0	10	0	10

	0	1	2	3	4
d:		10	20	30	40
p:		0	1	0	5
q:	0	10	0	1	0