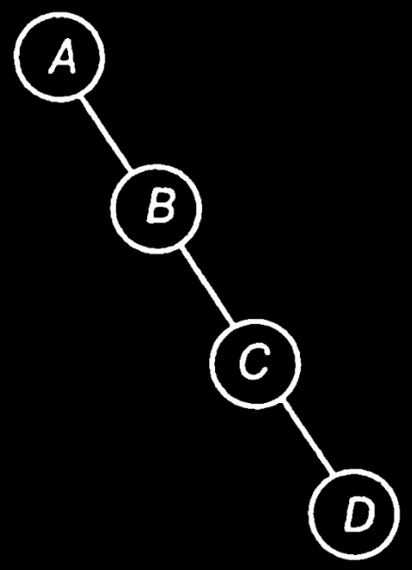
Оптимальные бинарные деревья поиска

Алгоритмы 9

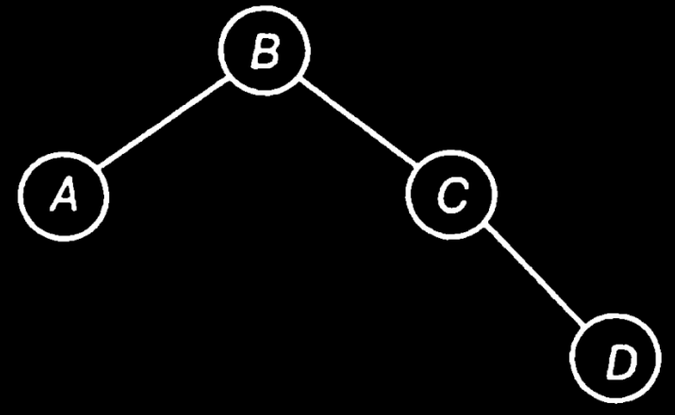
Бинарное дерево поиска

- Реализация словаря (поиск, вставка, удаление);
- Оптимальное дерево использует информацию о вероятности поиска каждого ключа.

Ключ	Вероятность
A	0,1
B	0,2
C	0,4
D	0,3



0,1*1+0,2*2+0,4*3+0,3*4 = **2,9**



0,2*1+0,1*2+0,4*2+0,3*3 = **2,1**

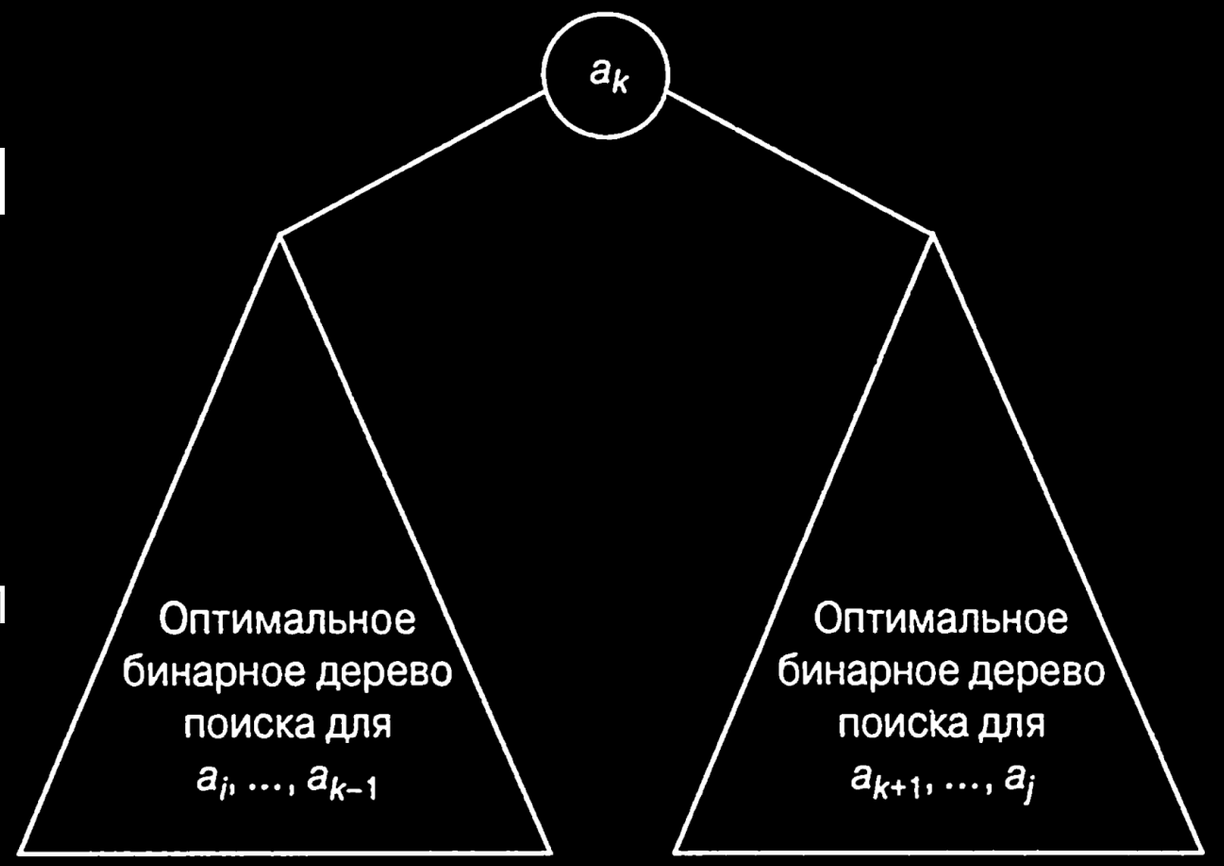
Каково оптимальное число сравнений?

Поддерево T_i^j

- $a_1, \ldots, a_n$ - различные ключи, упорядоченные по возрастанию;
- $p_1, \ldots, p_n$ - вероятности поиска этих ключей;
- C[i,j] - наименьшее среднее кол-во сравнений при успешном поиске в бинарном дереве T_i^j
- T_i^j состоит из $a_i, \ldots, a_j$, $1 <= i <= j <= n$.

Алгоритм поиска оптимального дерева

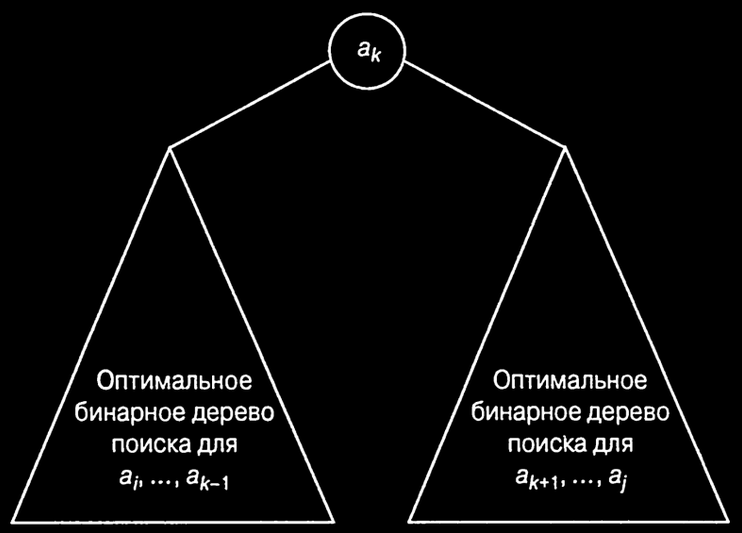
- **C[1,n]** - и есть искомое значение глубины
- **C[1,n]** вычисляем рекурсивно через рекуррентное соотношение для **C[i,j]**
- Если выбрать корень a_k,то
  - Левое поддерево содержит оптимально упорядоченные ключи $a_i, \dots, a_{k-1}$
  - Правое поддерево содержит оптимально упорядоченные ключи $a_{k+1}, \dots, a_j$



Рекуррентное соотношение

$$C[i,j] = \min_{i \leqslant k \leqslant j} \left\{ p_k \cdot 1 + \sum_{s=i}^{k-1} p_s \cdot \Big(\text{Уровень } a_s \text{ в } T_i^{k-1} + 1\Big) + \right.$$

$$\left. + \sum_{s=k+1}^{j} p_s \cdot \Big(\text{Уровень } a_s \text{ в } T_{k+1}^{j} + 1\Big) \right\} =$$



$$= \min_{i \leqslant k \leqslant j} \left\{ p_k + \sum_{s=i}^{k-1} p_s \cdot \text{Уровень } a_s \text{ в } T_i^{k-1} + \sum_{s=1}^{k-1} p_s + \right.$$

$$\left. + \sum_{s=k+1}^{j} p_s \cdot \text{Уровень } a_s \text{ в } T_{k+1}^{j} + \sum_{s=k+1}^{j} p_s \right\} =$$

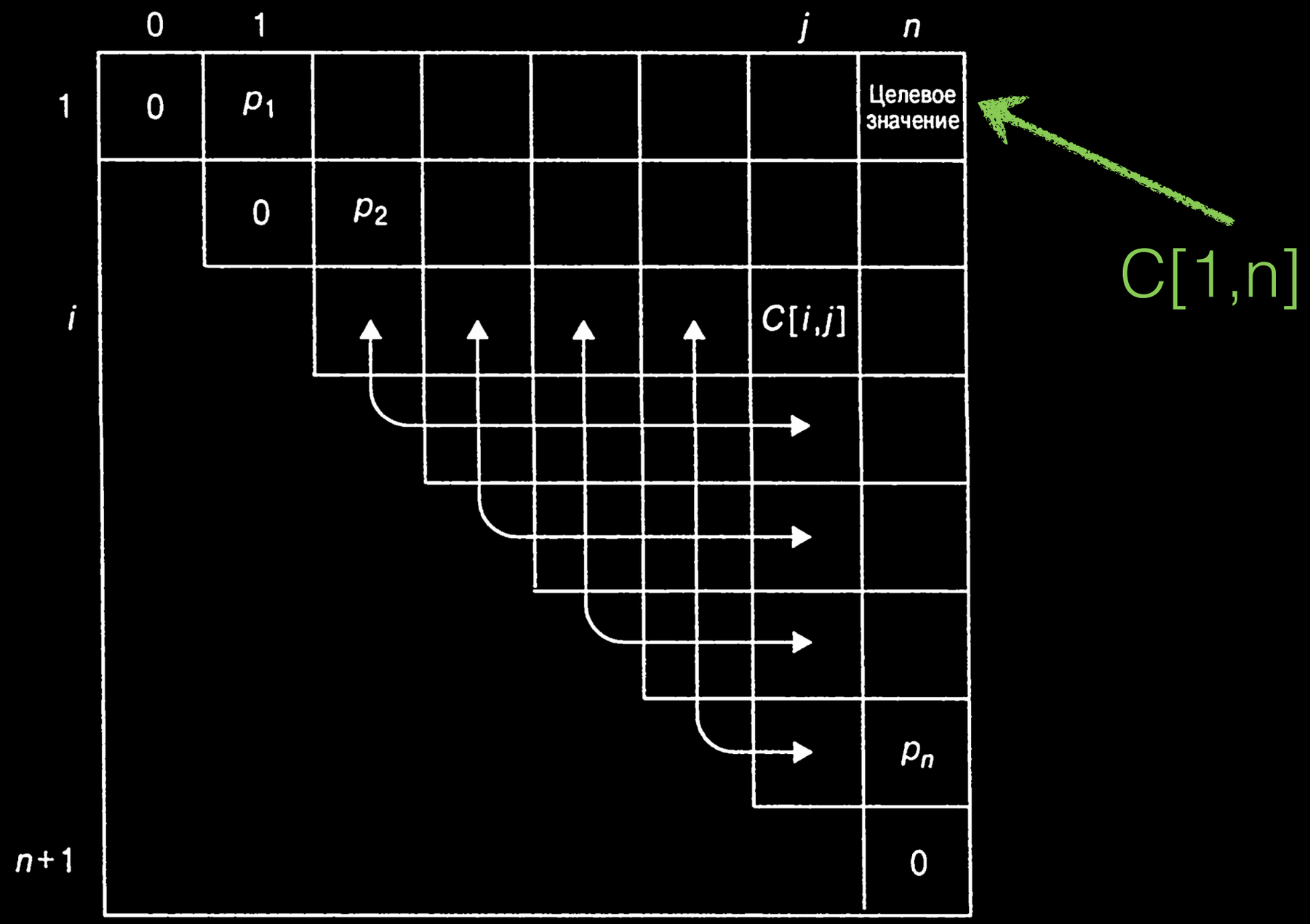
$$= \min_{i \leqslant k \leqslant j} \left\{ \sum_{s=i}^{k-1} p_s \cdot \text{Уровень } a_s \text{ в } T_i^{k-1} + \right.$$

Полагаем C[i,i-1] = 0

$$\left. + \sum_{s=k+1}^{j} p_s \cdot \text{Уровень } a_s \text{ в } T_{k+1}^{j} + \sum_{s=i}^{j} p_s \right\} =$$

$$= \min_{i \leqslant k \leqslant j} \{C[i,k-1] + C[k+1,j]\} + \sum_{s=i}^{j} p_s.$$

Значения, необходимые для вычисления C[i,j]



$$C[i,j] = \min_{i \leqslant k \leqslant j} \{C[i,k-1] + C[k+1,j]\} + \sum_{s=i}^{j} p_s.$$

Итак, мы вычислили среднее количество сравнений C[1,n]

А дерево-то как строим?

Нужно строить еще одну таблицу!

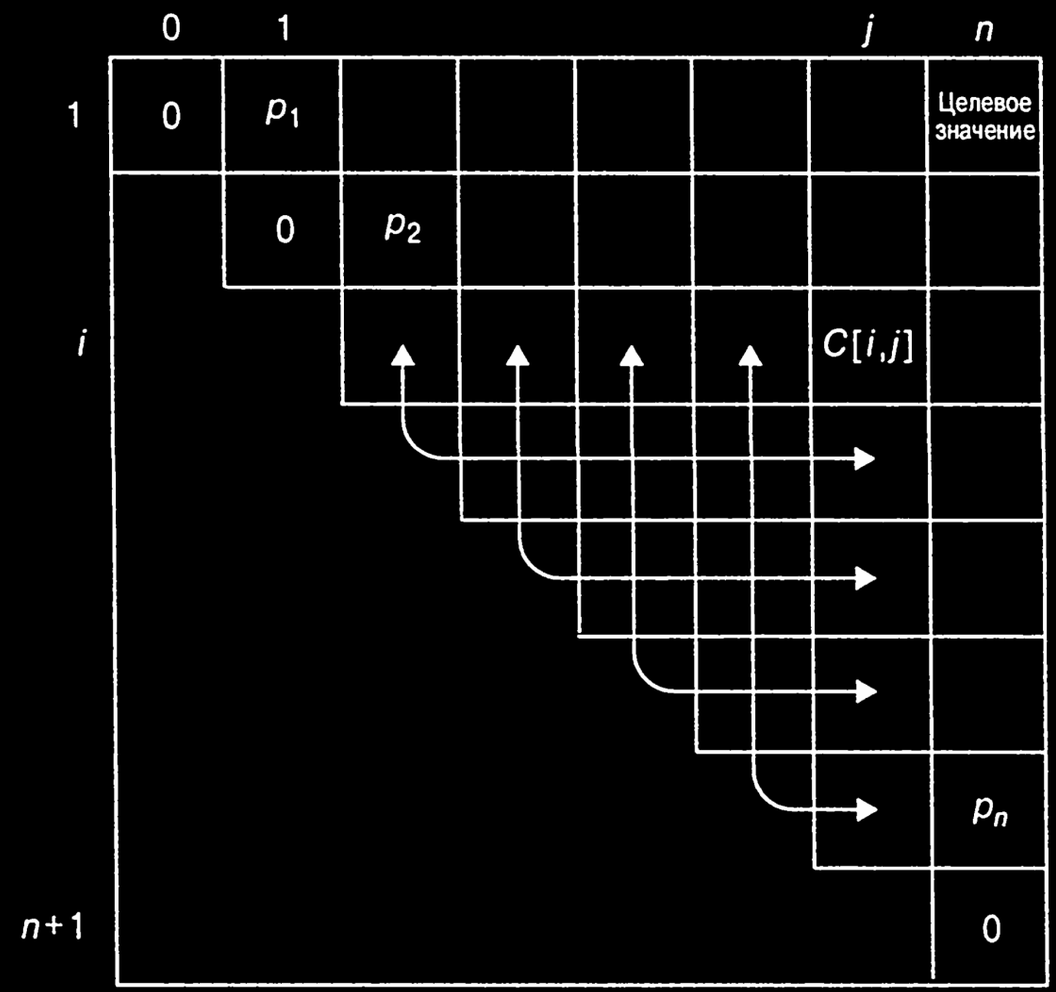
- В эту таблицу **R** записываем значения $\mathbf{R_{ij} = k}$, при которых достигается минимальное число проверок **C[i,j]**.

- Элементы этой таблицы являются индексами корней оптимальных поддеревьев!

- Из корней оптимальных поддеревьев строим оптимальное дерево.

Пример

Ключ	Вероятность
A	0,1
B	0,2
C	0,4
D	0,3



Главная таблица

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

Таблица корней

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

Ключ	Вероятность
A	0,1
B	0,2
C	0,4
D	0,3

Главная таблица

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

Таблица корней

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

$$C[1,2] = \min \begin{matrix} k=1: \; C[1,0] + C[2,2] + \sum_{s=1}^{2} p_s = 0 + 0.2 + 0.3 = 0.5 \\ k=2: \; C[1,1] + C[3,2] + \sum_{s=1}^{2} p_s = 0.1 + 0 + 0.3 = 0.4 \end{matrix} = 0.4.$$

Главная таблица

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

Таблица корней

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

Результат

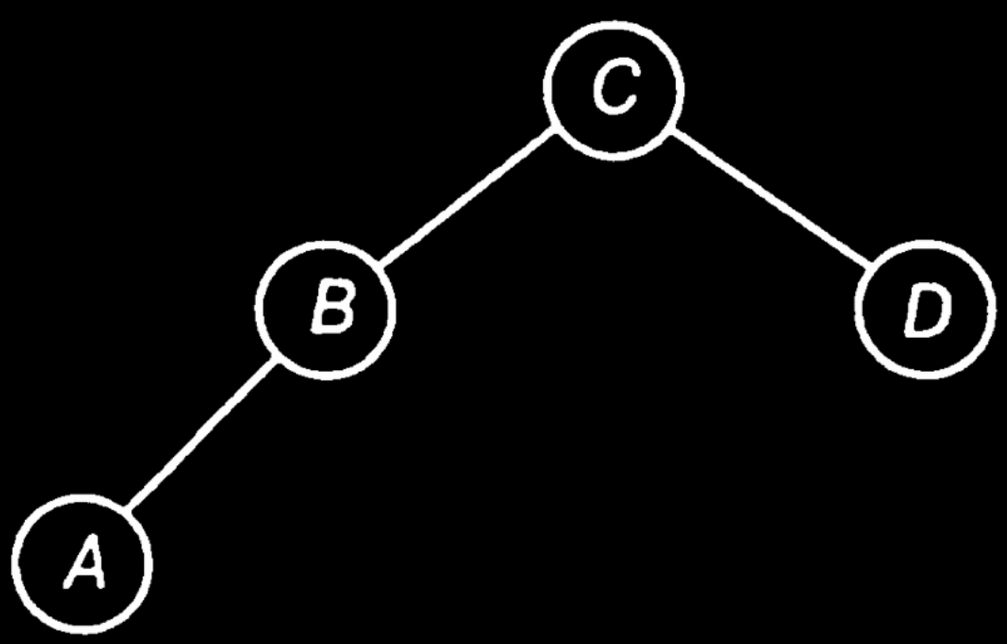
Ключ	Вероятнос
A	0,1
B	0,2
C	0,4
D	0,3

Главная таблица

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

Таблица корней

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					



Оптимальное дерево

R[1,2] = 2,
R[2,3] = 3,
R[3,4] = 3,
R[2,4] = 3,

Вопросы

- Почему алгоритму нужна квадратичная память
- Кубическое время?
- Как снизить время работы алгоритма до квадратичного?

Элементы таблицы корней располагаются всегда а неубывающем порядке и вдоль строк, и вдоль столбцов. Следовательно, R[i,j] ограничены диапазоном R[i,j-1], ... , R[i+1, j]. Как это использовать, чтобы снизить время до квадратичного?

Задача

- Реализуйте алгоритм построения оптимального бинарного дерева поиска.

Домашнее задание

1. Закончите построение дерева.
2. (*) Реализуйте алгоритм построения оптимального бинарного дерева поиска с квадратичным временем работы.