

# Деревья оптимального поиска (ДОП)

До сих пор предполагалось, что *все вершины дерева ищутся одинаково часто.*

Однако встречаются ситуации, когда **известны вероятности обращения к отдельным ключам дерева.**

Обычно для таких ситуаций характерно постоянство ключей (структура дерева остается неизменной).

Типичный пример - *сканер*

*компилятора*, который определяет, относится ли каждое слово программы (идентификатор) к **классу ключевых слов**.

Статистические измерения на сотнях компилируемых программ могут дать информацию об **относительных частотах** появления в тексте программы конкретных ключевых слов.

Припишем каждой вершине дерева  $V_i$  вес  $W_i$ , пропорциональный частоте поиска этой вершины.

Пример. Если из каждых **100 операций** поиска **15 операций** приходится на вершину  $V_1$ , то  $W_1 = 15$ .

**Сумма весов всех вершин** дает вес дерева **W**:

$$\mathbf{W} = \sum_{i=1}^n W_i$$

Каждая вершина  $V_i$  расположена на уровне  $h_i$ , корень дерева – на уровне 1.

Уровень вершины  $h_i$  - **количество операций сравнения**, необходимых для поиска данной вершины.

**Определение.** В дереве с  $n$  вершинами **средневзвешенная высота** равна

$$C_{\text{ср}} = h_{\text{срв}} = \frac{W_1 h_1 + W_2 h_2 + \dots + W_n h_n}{\mathbf{W}}$$

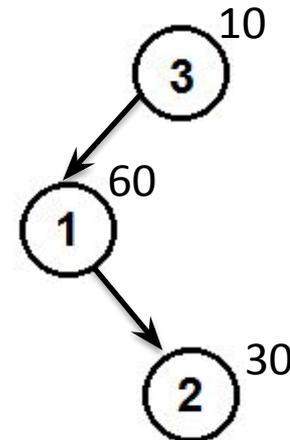
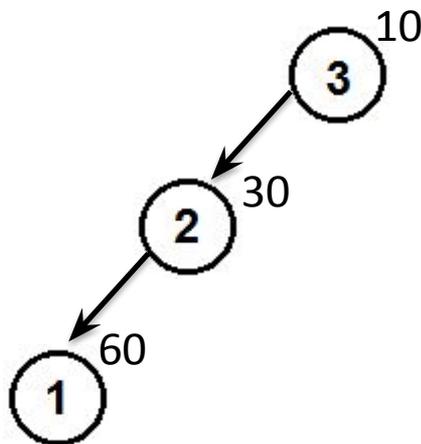
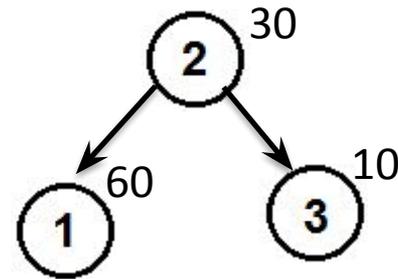
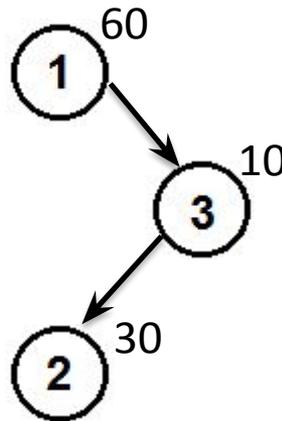
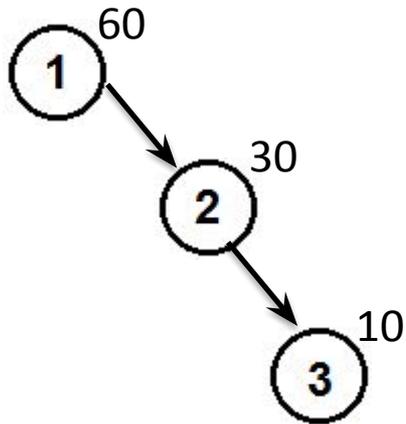
**Определение.** Дерево, имеющее **минимальную средневзвешенную высоту**, называется деревом оптимального поиска **ДОП.**

**Пример:** Рассмотрим множество из трех ключей

$$V_1 = 1, \quad V_2 = 2, \quad V_3 = 3;$$

$$W_1 = 60, \quad W_2 = 30, \quad W_3 = 10; \quad \mathbf{W} = 100.$$

Эти три ключа можно расположить в дереве поиска пятью различными способами:



$$1. h_{\text{срв}} = \frac{60*1+30*2+10*3}{100} = 1,5$$

$$2. h_{\text{срв}} = \frac{60*1+10*2+30*3}{100} = 1,7$$

$$3. h_{\text{срв}} = \frac{30*1+60*2+10*2}{100} = 1,7$$

$$4. h_{\text{срв}} = \frac{10*1+30*2+60*3}{100} = 2,5$$

$$5. h_{\text{срв}} = \frac{10*1+60*2+30*3}{100} = 2,2$$

Очевидно, для минимизации средней длины пути поиска нужно стремиться **вершины с наибольшим весом располагать ближе к корню дерева.**

## Задача построения ДОП

может ставиться в двух вариантах:

1. Известны вершины и их веса (дерево не меняется в процессе поиска).
2. Вес вершины определяется в процессе работы (после каждого поиска вершины её вес увеличивается на единицу).

В этом случае нужно перестраивать структуру дерева.

- **Точный алгоритм построения ДОП**

3 вершины = 5 конфигураций дерева

4 вершины = 14 конфигураций дерева

5 вершин = 62 конфигурации дерева

**Количество возможных конфигураций дерева из  $n$  вершин *с ростом  $n$  растет экспоненциально*, т.е. как  $e^n$ .**

**Таким образом, при больших  $n$  задача построения ДОП кажется безнадёжной.**

Однако оптимальные деревья обладают важным свойством, которое помогает их обнаруживать:

*все их поддеревья также оптимальны.*

На этом свойстве основан алгоритм, который находит все большие и большие оптимальные поддеревья, начиная с отдельных вершин.

Т.к. вес дерева постоянный, то вместо средневзвешенной высоты будем рассматривать **взвешенную высоту дерева**:

$$P = W_1 h_1 + W_2 h_2 + \dots + W_n h_n$$

Для ДОП справедливо соотношение:

$$P = P_L + \mathbf{W} + P_R,$$

где  $P_L, P_R$  — взвешенные высоты левого и правого поддеревьев.

Обозначим

$T_{ij}$  — оптимальное поддерево,

состоящее из вершин  $V_{i+1}, \dots, V_j$

$W_{ij}$  — вес поддерева  $T_{ij}$        $P_{ij}$  — взвешенная высота  $T_{ij}$

Очевидно,

$P = P_{0,n}$  — взвешенная высота всего дерева из  $n$  вершин

$\mathbf{W} = W_{0,n}$  — вес всего дерева

$T_{ii}$  — пустое поддерево  $\rightarrow W_{i,i} = 0, P_{i,i} = 0.$

Величины  $W_{ij}$  и  $P_{ij}$  вычисляются по рекуррентным формулам для всех возможных поддеревьев:

$$(1) \quad W_{ij} = W_{i,j-1} + W_j \quad 0 \leq i < j \leq n$$

$$(2) \quad P_{ij} = W_{ij} + \min_{i < k \leq j} (P_{i,k-1} + P_{k,j}) \quad 0 \leq i < j \leq n$$

При нахождении **min** будем запоминать **k**, при котором достигается **min**, обозначать его **k\*** и записывать в матрицу  $r_{ij}$ .

Это значение является **индексом (номером) корня поддерева  $T_{ij}$  в упорядоченном массиве вершин.**

№	1	2	3			...			n
k	$k_1$	$k_2$	$k_3$			...			$k_n$

**Матрица корней поддеревьев  $r_{ij}$  полностью определяет структуру дерева.**

**Пример:**

$V_1=1, V_2=2, V_3=3;$

$W_1=60, W_2=30, W_3=10; \mathbf{W} = 100$

$T_{00}, T_{11}, T_{22}, T_{33}$  - пустые поддеревья

$T_{01}, T_{12}, T_{23}$  - поддеревья из одной вершины

$T_{02}, T_{13}$  - поддеревья из двух вершин

$T_{03}$  - дерево из трех вершин

	0	1	2	3
0	0	60	90	100
1		0	30	40
2			0	10
3				0

Пример.  $V_1=1, V_2=2, V_3=3;$

$W_1=60, W_2=30, W_3=10; \mathbf{W} = 100$

**Вычисление взвешенных высот поддеревьев:**

$$P_{01} = W_{01} + \min_{0 < k \leq 1} (P_{00} + P_{11}) = 60 \quad \mathbf{k^*=1}$$

$$P_{12} = W_{12} + \min_{1 < k \leq 2} (P_{11} + P_{22}) = 30 \quad \mathbf{k^*=2}$$

$$P_{23} = W_{23} + \min_{2 < k \leq 3} (P_{22} + P_{33}) = 10 \quad \mathbf{k^*=3}$$

	0	1	2	3
0	0	60		
1		0	30	
2			0	10
3				0

$$P_{02} = W_{02} + \min_{0 < k \leq 2} (P_{00} + P_{12}, P_{01} + P_{22}) =$$

$$= 90 + \min (0+30, 60+0) = 120 \quad \mathbf{k^*=1}$$

$$P_{13} = W_{13} + \min_{1 < k \leq 3} (P_{11} + P_{23}, P_{12} + P_{33}) =$$

$$= 40 + \min (0+10, 30+0) = 50 \quad \mathbf{k^*=2}$$

$$P_{03} = W_{03} + \min_{0 < k \leq 3} (P_{00} + P_{13}, P_{01} + P_{23}, P_{02} + P_{33}) =$$

$$= 100 + \min (0+50, 60+10, 120+0) = 150 \quad \mathbf{k^*=1}$$

	0	1	2	3
0	0	60	120	150
1		0	30	50
2			0	10
3				0

	0	1	2	3
0	0	1	1	1
1		0	2	2
2			0	3
3				0

Идея алгоритма построения ДОП по матрице  $r_{ij}$

Исходные вершины должны быть **упорядочены по ключам**, в матрице берем  $r_{0,n}$  — это номер корня всего дерева в упорядоченном массиве вершин.

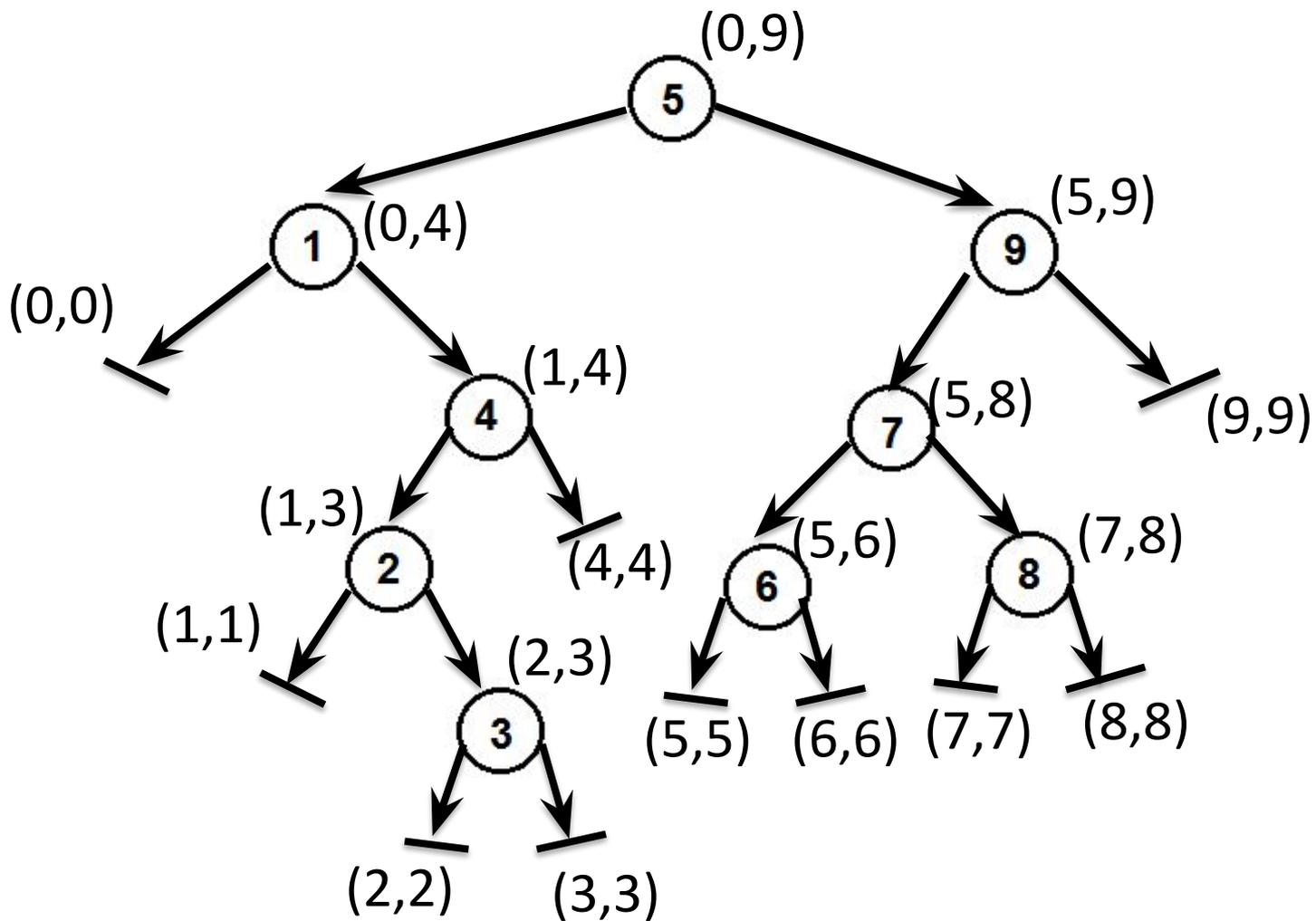
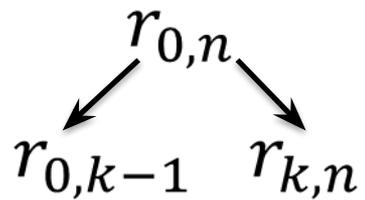
Добавляем эту вершину в дерево через обычное добавление в СДП.

Затем в матрице  $r_{ij}$  берем значение  $r_{0,k-1}$  ( $k=r_{0,n}$ ) и добавляем эту вершину в левое поддерево, берем  $r_{k,n}$  и добавляем вершину в правое поддерево и т.д.

**Пример:** Возьмем произвольную матрицу  $r_{ij}$  для 9 вершин и построим по ней ДОП.

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	1	4	4	5	5	5
1		0	2	2	4	5	5	5	5	7
2			0	3	4	5	5	5	5	7
3				0	4	5	5	5	5	7
4					0	5	5	5	7	7
5						0	6	7	7	9
6							0	7	7	9
7								0	8	9
8									0	9
9										0

Шаг 1. Проверка матрицы на корректность



## Трудоёмкость метода:

Существует  $\frac{n^2}{2}$  значений  $P_{ij}$ , формула (2) требует выбора одного из  $0 \leq i < j \leq n$  значений, трудоёмкость сводится к  $\frac{n^3}{6}$  операций, т.е.

**трудоёмкость кубическая.**

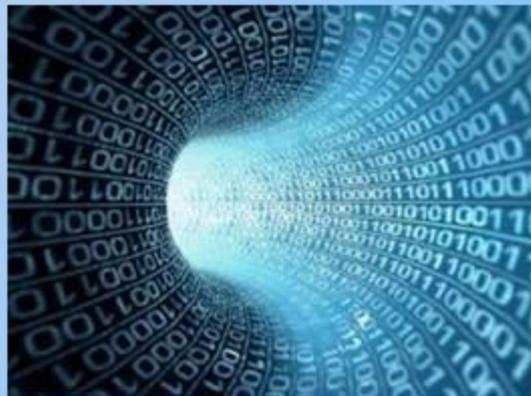
В матрице существует закономерность

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

Это позволяет сократить поиск  $r_{ij}$  до этого диапазона, что дает возможность уменьшить трудоёмкость до **квадратичной**.

# Программист должен ПОМНИТЬ

Компьютер не признает никаких  
интуитивных выводов и предположений  
со слов "кажется".



Постоянно гореть работой, изобретать —  
такой выбор творческих людей.

# Алгоритм построения ДОП

**Вычисление  $AW$  - матрицы весов:**

DO (i=0, n)

DO (j=i+1, n)

$$AW_{ij} = AW_{ij-1} + W_i$$

OD

OD

**Вычисление матриц  $AP$  и  $AR$ :**

Обозначим  $h = j - i$  – размер поддеревя

**При  $h = 1$ :**

DO (i=0, n-1)

$$j=i+1; AP_{ij} = AW_{ij}; AR_{ij} = j$$

OD

**При  $h > 1$ :**

DO (h=2,3,...,n) ...

# Алгоритм построения ДОП

При  $h > 1$ :  $h = j - i$  – размер поддеревя

```
DO ( h = 2, 3, ..., n )
  DO ( i = 0, ..., n - h )
    j := i + h
    m := ARi, j-1
    min := APi, m-1 + APm, j
      DO ( k = m+1, ..., ARi+1, j )
        x := APi, k-1 + APk, j
        IF ( x < min ) m := k , min := x FI
      OD
    APi, j := min + AWi, j
    ARi, j := m
  OD
OD
```

## Создание дерева: Create Tree(0,n)

Create Tree(L,R) // L,R - границы массива вершин V

IF(L<R)

k = AR<sub>LR</sub>;

Добавить (Root, V<sub>k</sub>)

Create Tree(L,k-1)

Create Tree(k,R)

FI

Трудоёмкость точного алгоритма построения ДОП:

по времени  $O(n^2)$

по занимаемой памяти  $O(n^2)$

При больших объемах деревьев такие алгоритмы становятся неэффективными.



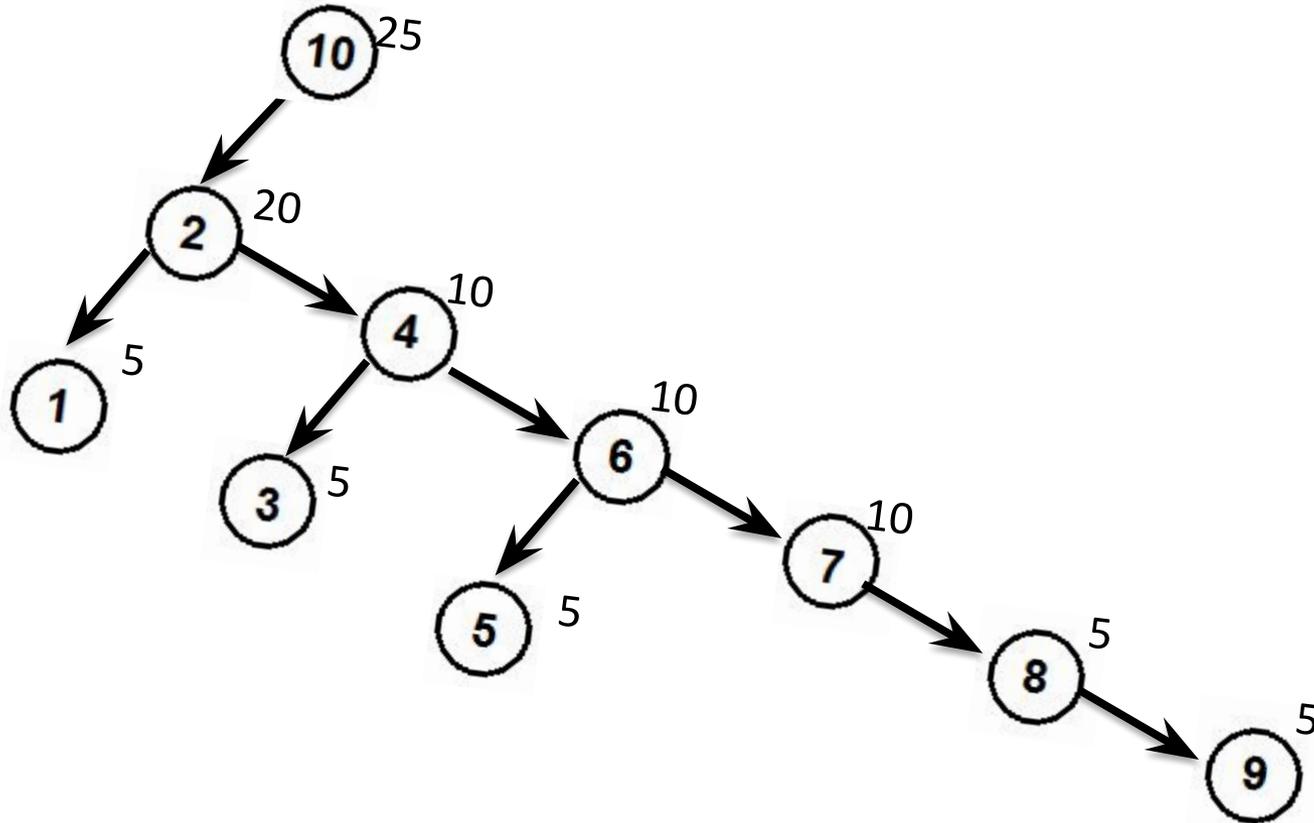
# Приближенные алгоритмы построения ДОП

Известны **быстрые** алгоритмы, строящие **почти оптимальные** деревья поиска.  
Назовем эти алгоритмы  $A_1$  и  $A_2$ .

Алгоритм  $A_1$ :

В качестве корня **берем вершину с наибольшим весом**, будем поступать так же для каждого поддеревя.

	1	2	3	4	5	6	7	8	9	10
	5	20	5	10	5	10	10	5	5	25



$$h = \frac{25 \cdot 1 + 20 \cdot 2 + 5 \cdot 3 + 10 \cdot 3 + 5 \cdot 4 + 10 \cdot 4 + 5 \cdot 5 + 10 \cdot 5 + 5 \cdot 6 + 5 \cdot 7}{100} = 3,1$$

- **Алгоритм A1 на псевдокоде**

<сортировка по убыванию весов>

DO (i=1, n)

Добавить (Root,  $V_i$ )

OD

## Алгоритм А2:

Алгоритм отличается тем, что **вершины должны быть упорядочены по возрастанию ключей.**

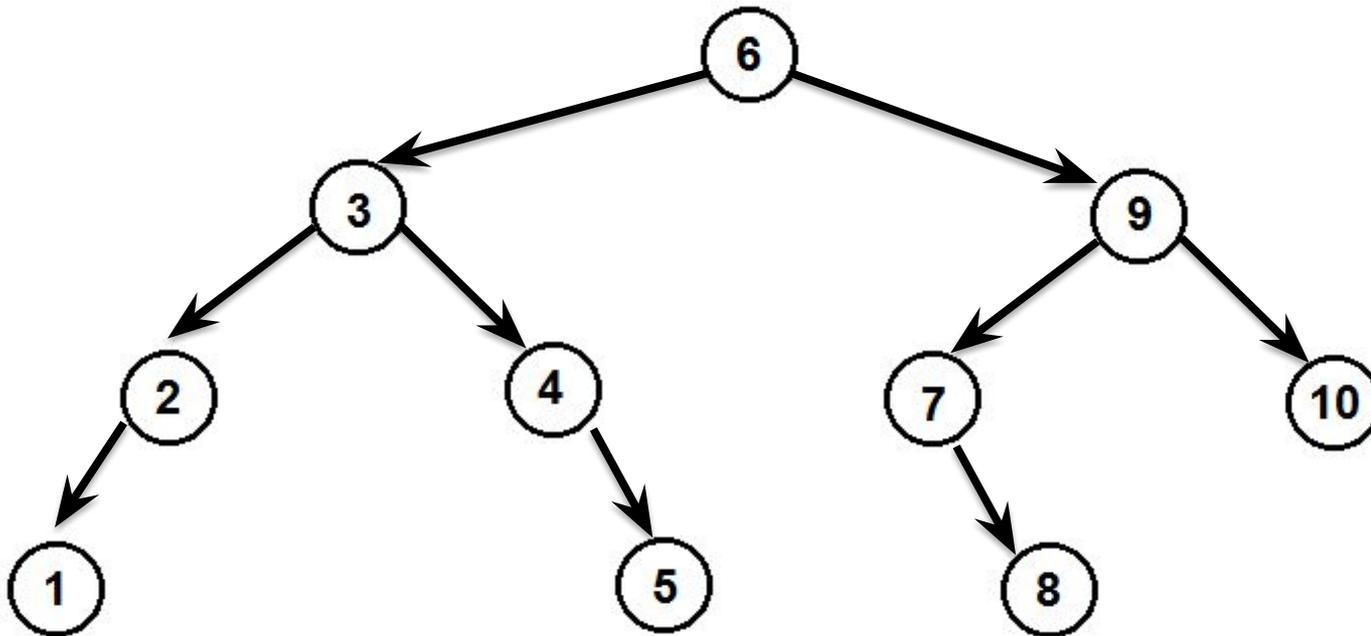
**Находим «центр тяжести»**, т.е. путем последовательного суммирования весов определим вершину  $V_k$ , для которой справедливы неравенства:

$$\sum_{i=1}^{k-1} w_i < \frac{W}{2} \quad \text{и} \quad \sum_{i=1}^k w_i \geq \frac{W}{2}$$

В качестве **«центра тяжести»** берется вершина, для которой сумма весов левой и правой части массива как можно меньше отличаются друг от друга.

Иногда для большей точности также рассматриваются две ближайшие к выбранной вершины:  $V_{k-1}$  и  $V_{k+1}$ .

	1	2	3	4	5	6	7	8	9	10
	5	20	5	10	5	10	10	5	5	25



$$h_{\text{cpB}} = \frac{10 \cdot 1 + 5 \cdot 2 + 5 \cdot 2 + 20 \cdot 3 + 10 \cdot 3 + 10 \cdot 3 + 10 \cdot 3 + 25 \cdot 3 + 5 \cdot 4 + 5 \cdot 4 + 5 \cdot 4}{100} = 2,85$$

- **Алгоритм A2 на псевдокоде**

**A2 (L, R)**

wes = 0, sum = 0

IF (L ≤ R)

DO ( i = L, L+1, ..., R ) wes = wes +  $W_i$  OD

DO( i = L, L+1, ..., R)

IF (sum < wes/2 и sum +  $W_i$  > wes/2 ) OD FI

sum = sum +  $W_i$

OD

Добавить ( root,  $V_i$ )

A2 ( L, i-1 )

A2 ( i+1, R)

FI

- **Трудоёмкость приближенных алгоритмов:**  
по времени  $O(n \cdot \log_2 n)$   
по занимаемой памяти  $O(n)$
- Алгоритм **A1** «плохой» : при  $n \rightarrow \infty$  дерево **равносильно случайному** (с точки зрения средневзвешенной высоты);
- Алгоритм **A2** хороший: дерево **асимптотически приближается к оптимальному.**

# КУРАПОВА ЕЛЕНА ВИКТОРОВ НА

	К	У	Р	А	П	О	В	Е	Л	Н	И	Т
	2	1	2	4	1	3	3	2	1	2	1	1

	А	В	Е	И	К	Л	Н	О	П	Р	Т	У
	4	3	2	1	2	1	2	3	1	2	1	1

**ДЗ:** Построить почти оптимальные деревья с помощью алгоритмов А1 и А2. Вычислить и сравнить средневзвешенные высоты полученных деревьев.

**В лаб.работе** для проверки правильности точного алгоритма сравнить:

$$\frac{AP[0, n]}{AW[0, n]} \text{ (из матриц) } = h_{\text{срв}} \text{ (по дереву)}$$

# Relax

