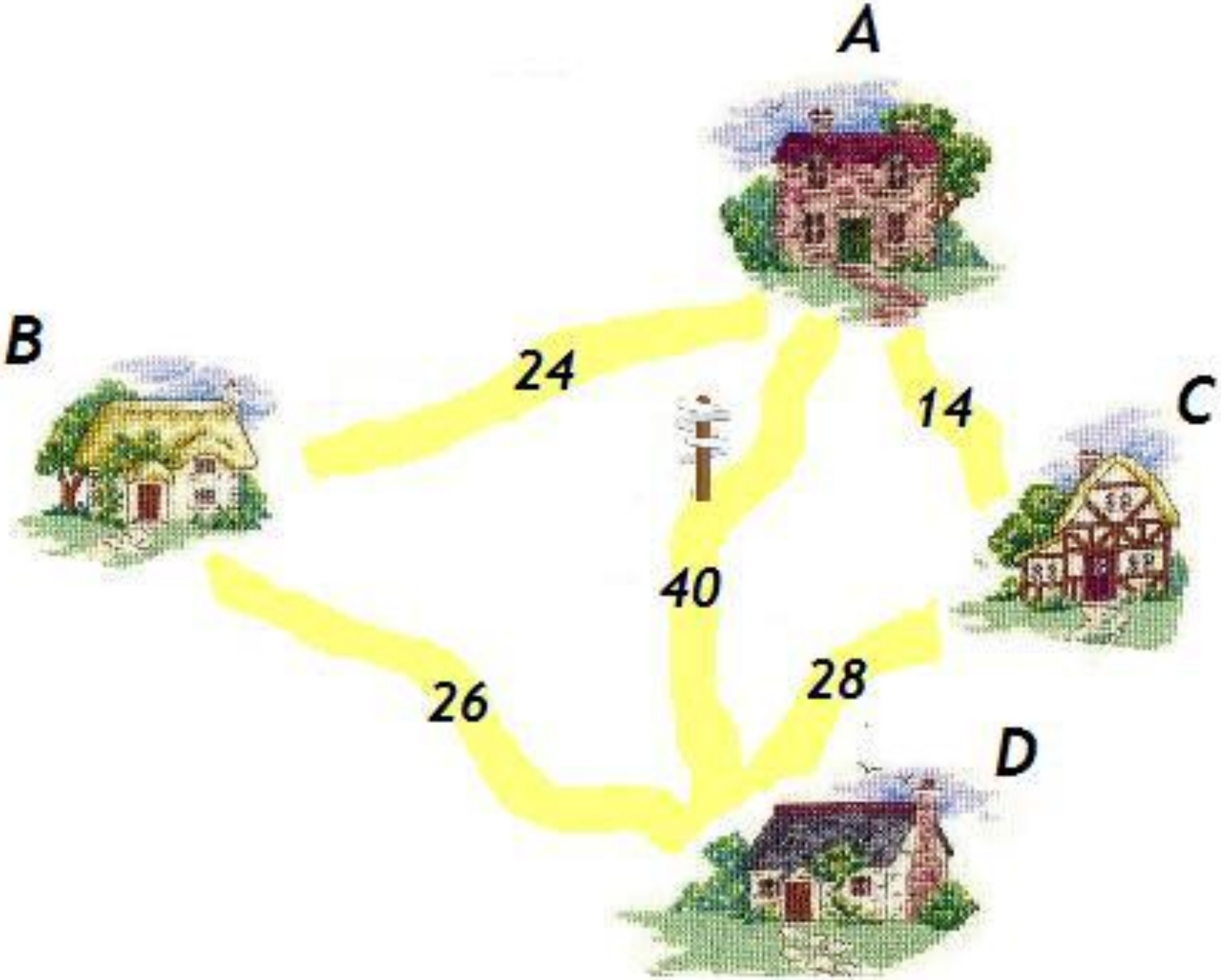
Лекция 7.
Применение связанных списков.

Применение связных списков

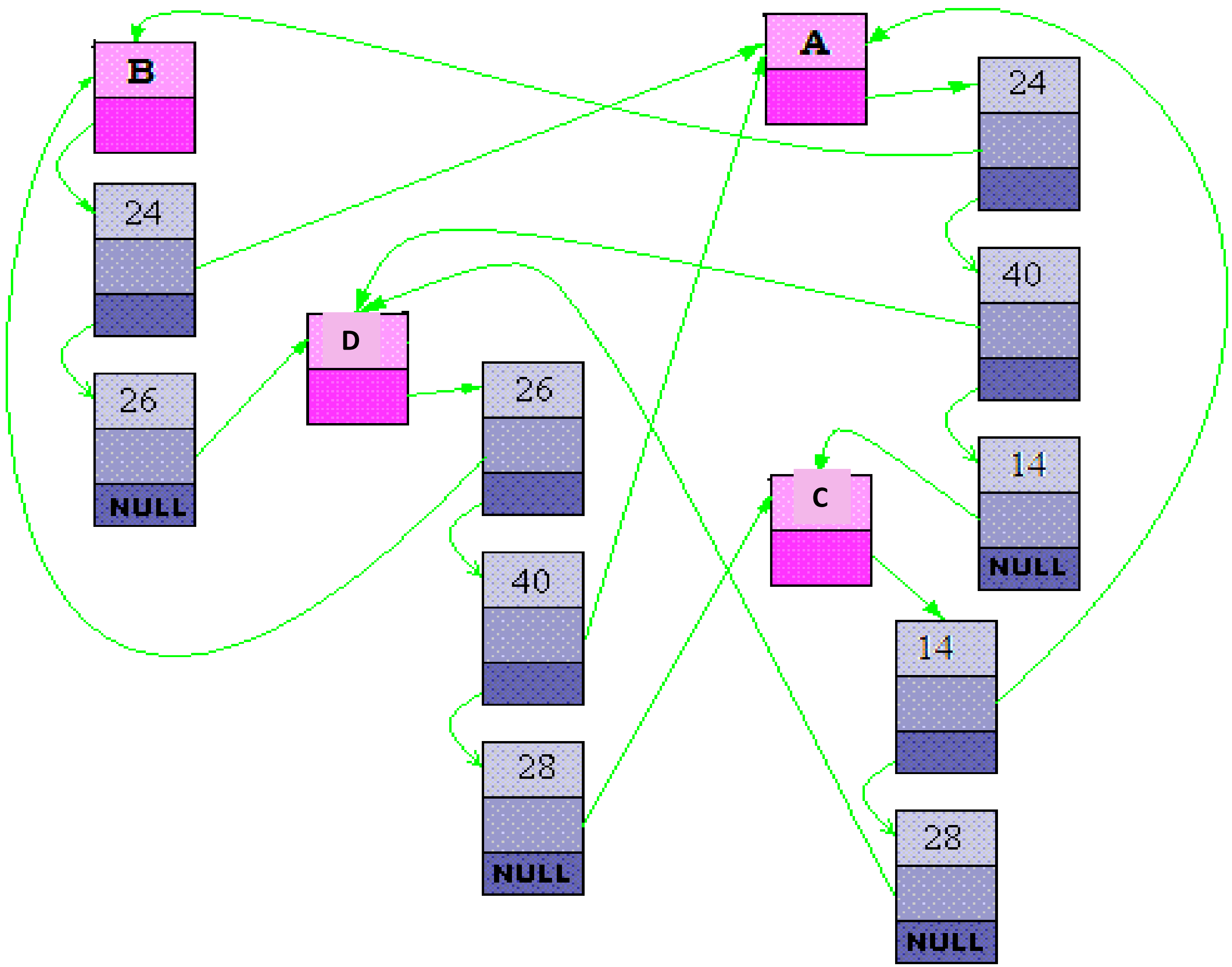
- Сложные структуры данных (карта дорог)
- Разреженные матрицы
- Сложение и умножение многочленов
- Сортировка и поиск (бинарные деревья поиска)
- Объектно-ориентированное программирование

Карта дорог

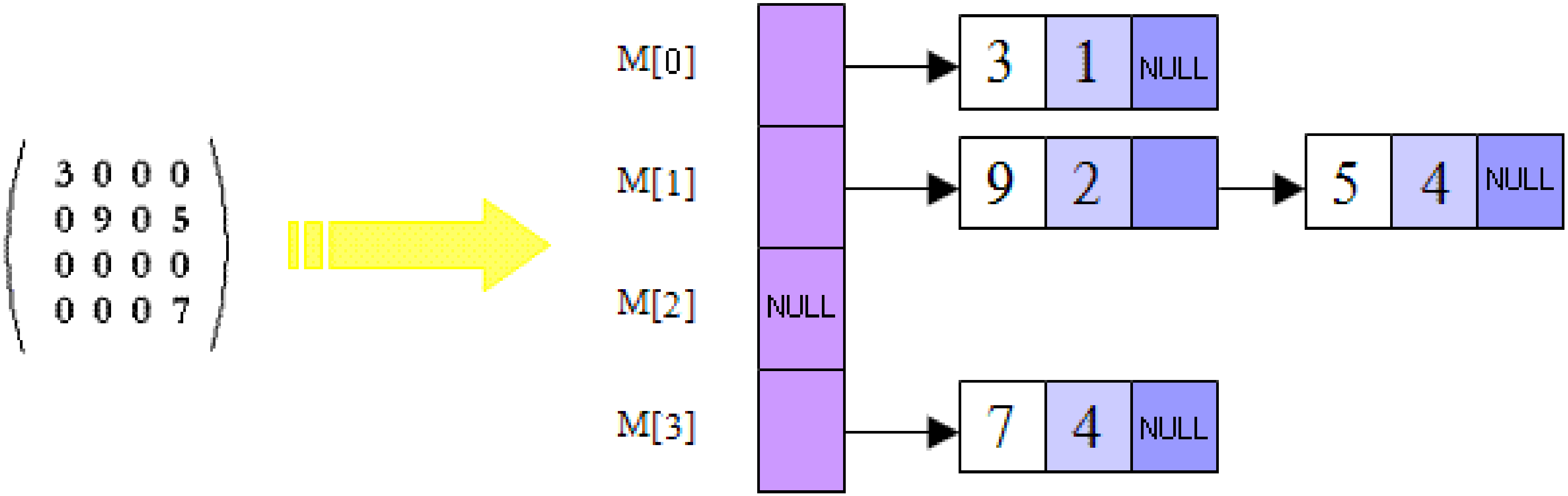
•



Карта дорог



Разреженные матрицы



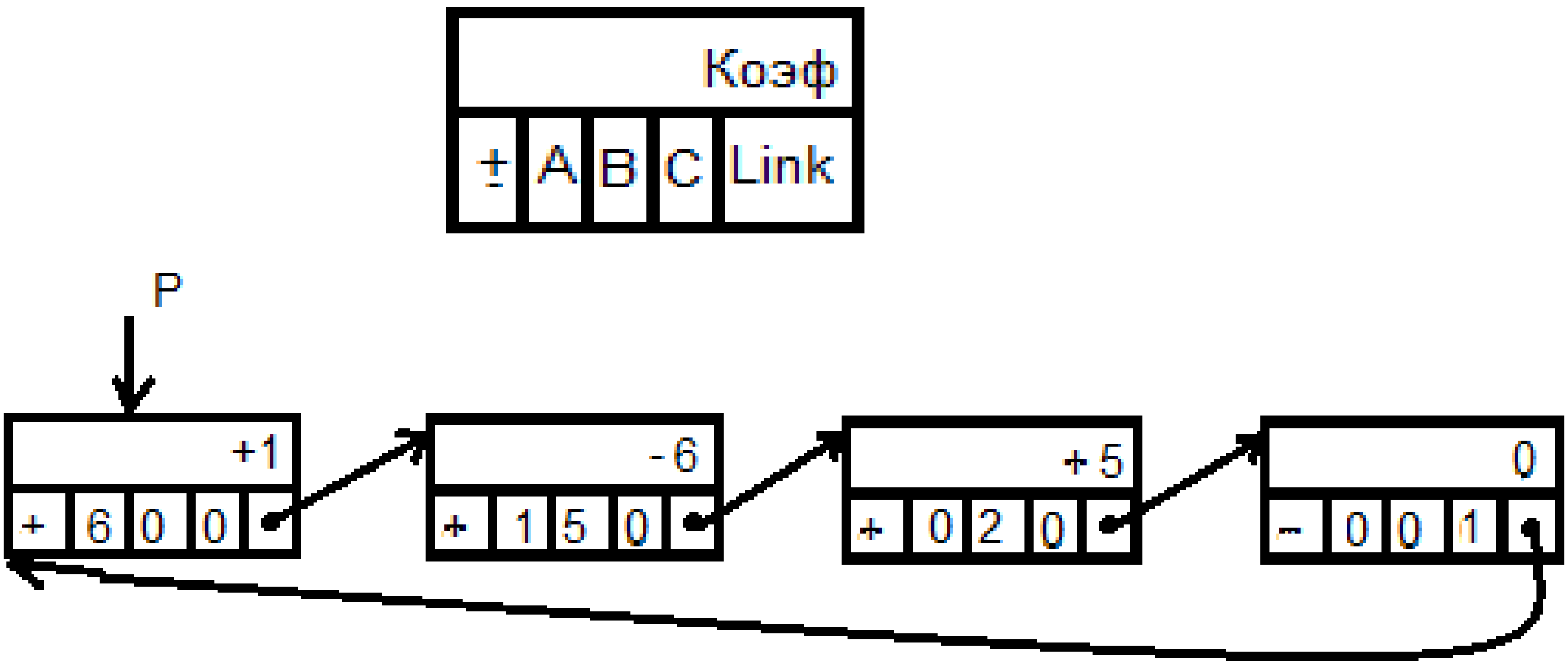
Сложение и умножение многочленов

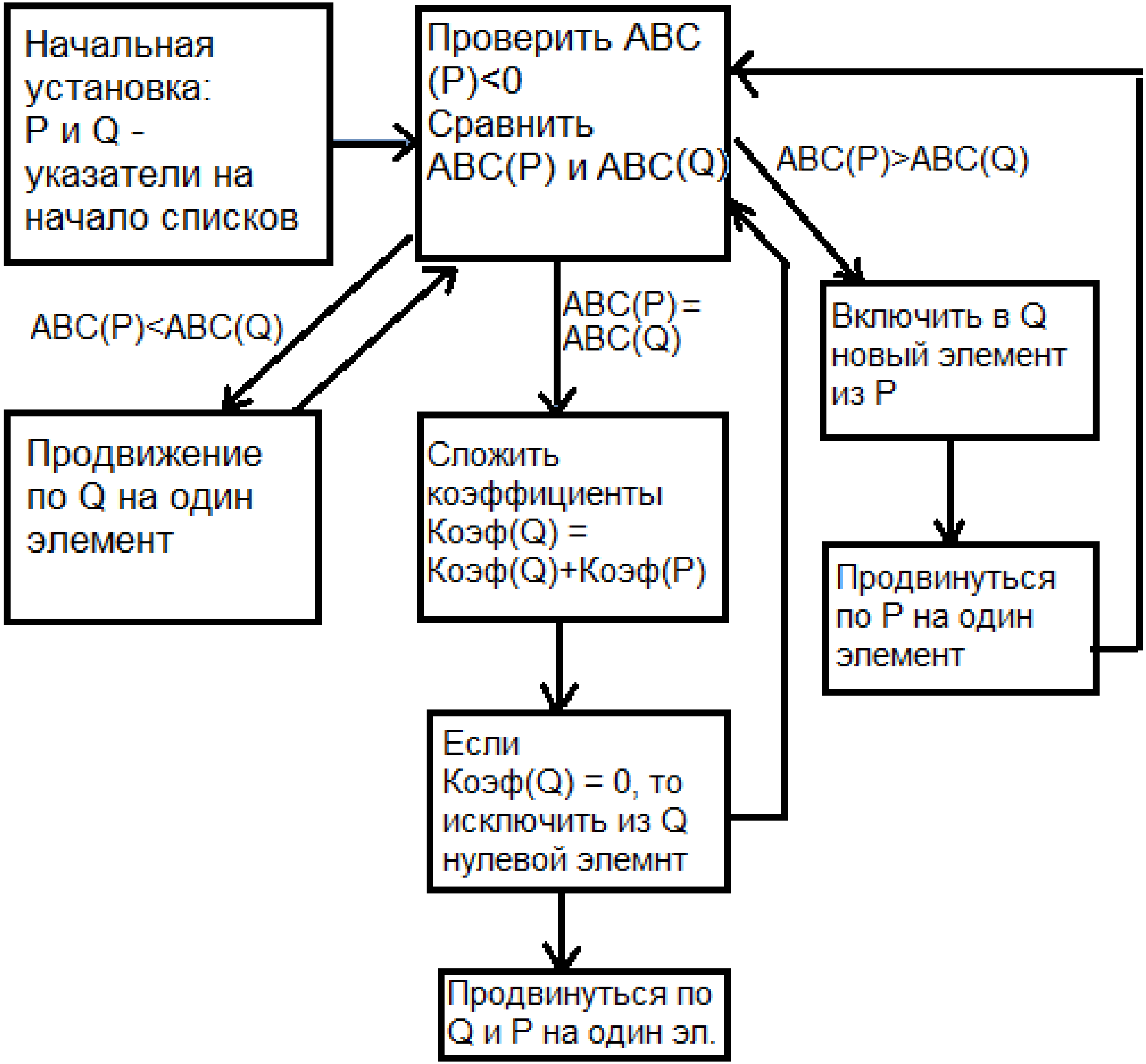
P: $x^6 - 6*x*y^5 + 5*y^2$

Q: $3*x^2*y + 4*x*y^5$

P+Q: $x^6 + 3*x^2*y - 2*x*y^5 + 5*y^2$

Сложение и умножение многочленов




Начальная установка: P и Q - указатели на начало списков
Проверить ABC (P)<0 Сравнить ABC(P) и ABC(Q)
ABC(P)>ABC(Q)
ABC(P)<ABC(Q)
ABC(P) = ABC(Q)
Включить в Q новый элемент из P
Продвижение по Q на один элемент
Сложить коэффициенты Коэф(Q) = Коэф(Q)+Коэф(P)
Продвинуться по P на один элемент
Если Коэф(Q) = 0, то исключить из Q нулевой элемнт
Продвинуться по Q и P на один эл.

Произвольные деревья

Дерево – конечное множество T, состоящее из одного или более узлов, таких, что:

а) имеется один специально обозначенный узел, называемый корнем;

б) остальные узлы содержатся в $m \geq 0$ попарно не пересекающихся множествах $T1, T2, \ldots, Tm$, каждое из которых в свою очередь является деревом.

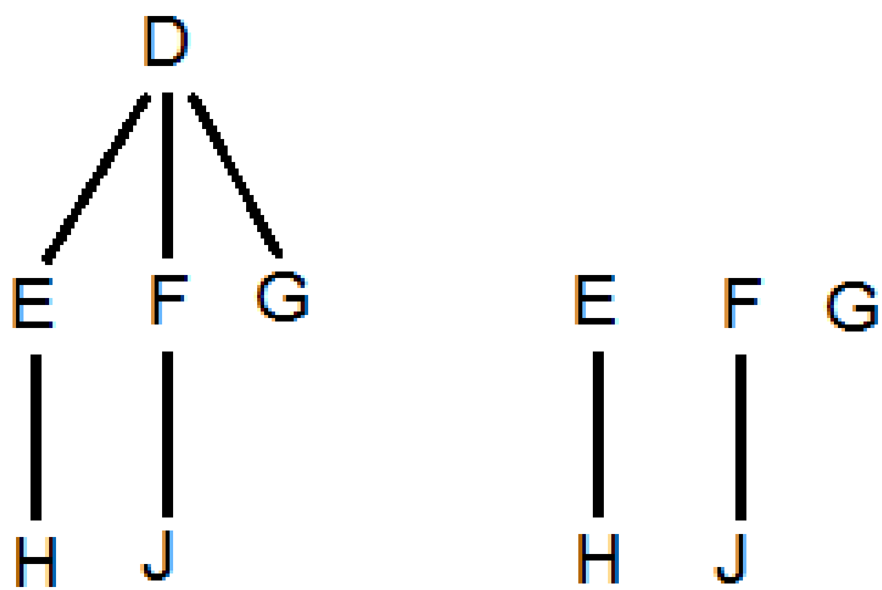
Бинарное дерево

Бинарное дерево – это конечное множество узлов, которое или пусто, или состоит из корня и двух непересекающихся бинарных деревьев, называемых левым и правым поддеревьями данного дерева.

Лес деревьев

Лес – это множество, состоящее из некоторого (быть может, равного нулю) числа непересекающихся деревьев.

Соответствие между абстрактными лесами и деревьями

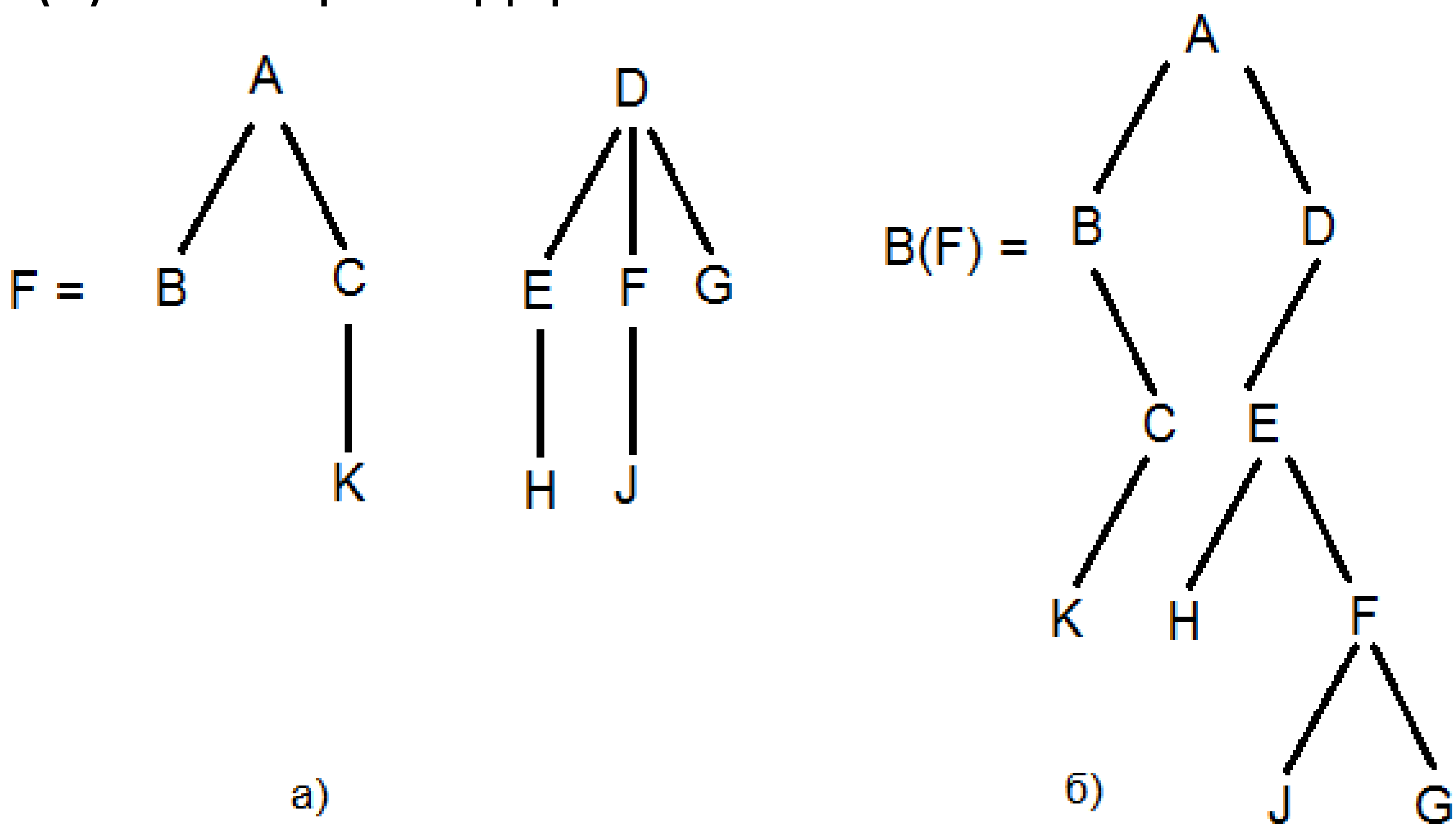


а)   б)

Соответствие между лесами и бинарными деревьями

F(T1, T2,…, Tn) - лес деревьев

B(F) – бинарное дерево

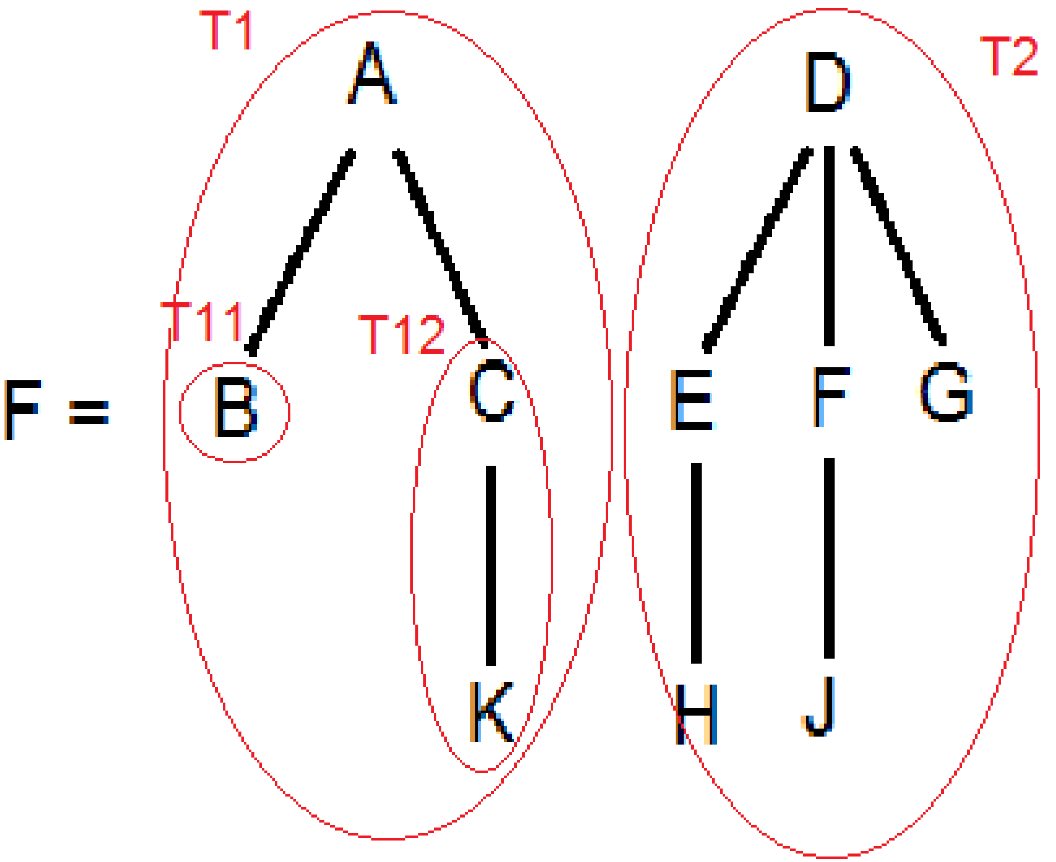


а)

б)

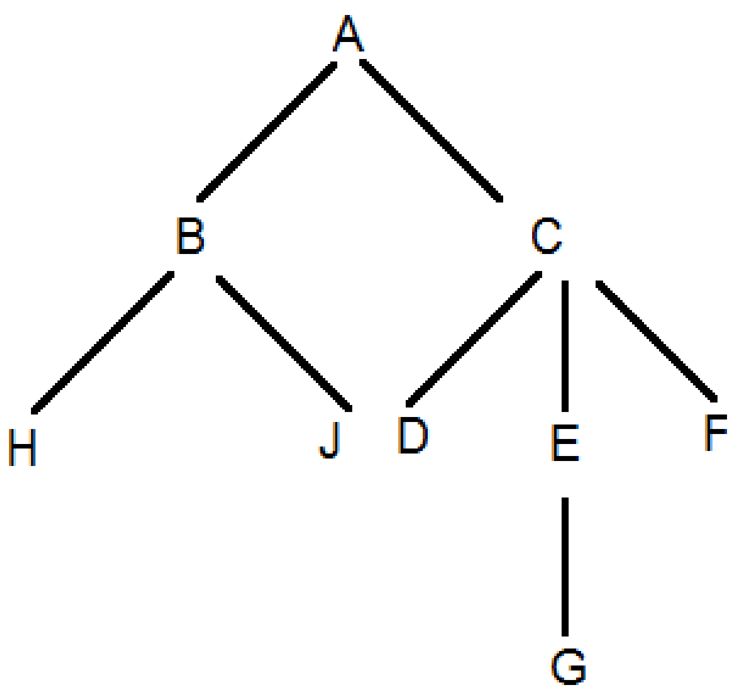
Если n=0, то B(F) пусто.

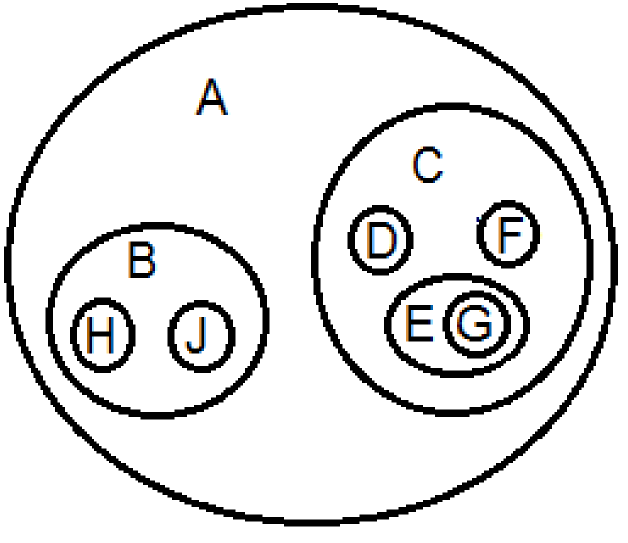
Если n>0, то корнем B(F) является корень дерева T1, левым поддеревом является B(T11, T12,…), правым поддеревом дерева B(F) является B(T2,…, Tn).



Другие представления деревьев

Вложенные множества



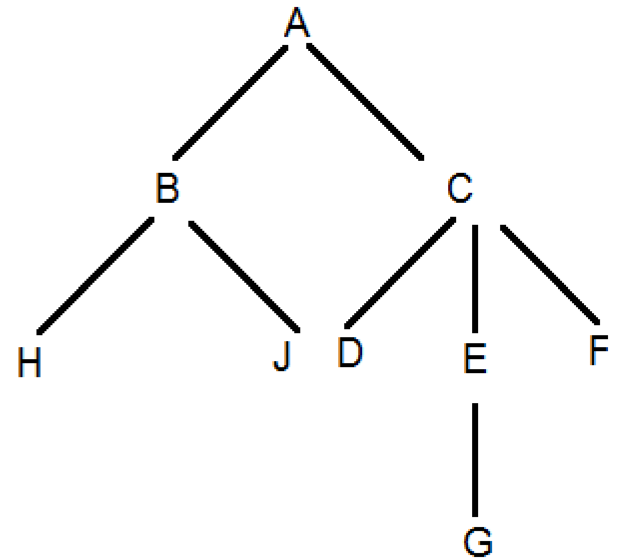


Другие представления деревьев

Десятичная система обозначений Дьюи

1A; 1.1B; 1.1.1H; 1.1.2J;

1.2C; 1.2.1D; 1.2.2E; 1.2.2.1G; 1.2.3F

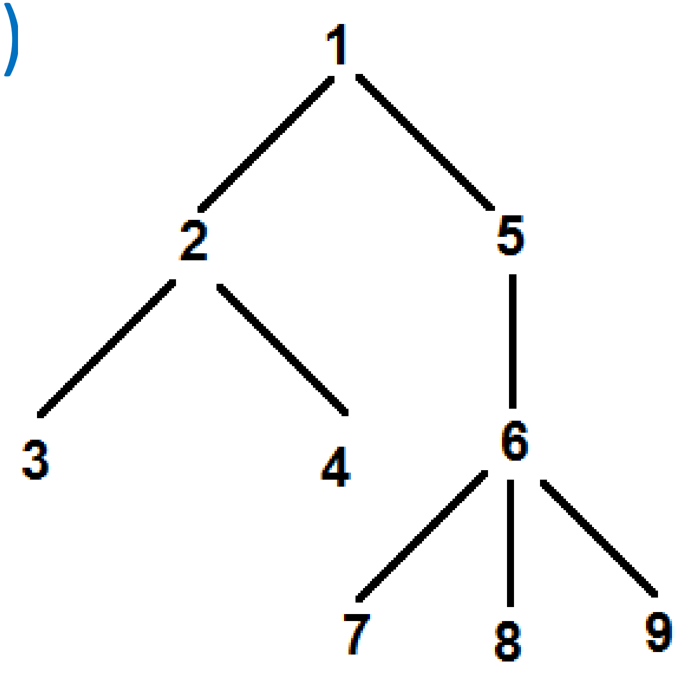


Скобочные представления деревьев

Левое скобочное представление

$L(T) = a(L(T1), L(T2), \ldots , L(Tk))$

$L(T) = 1(2(3,4), 5(6(7,8,9)))$

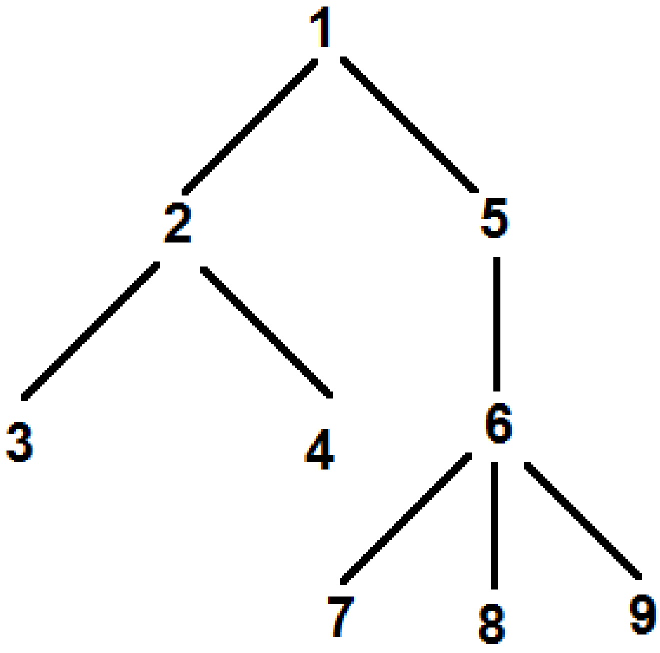


Скобочные представления деревьев

Правое скобочное представление

R(T) = (R(T1), R(T2), … , R(Tk))a

R(T) = ((3,4)2, ((7,8,9)6) 5)1



Представление алгебраических выражений с помощью бинарных деревьев

Выражение:

(a + b) * c

1. Левое скобочное представление:

*(+(a,b), c)

2. Правое скобочное представление

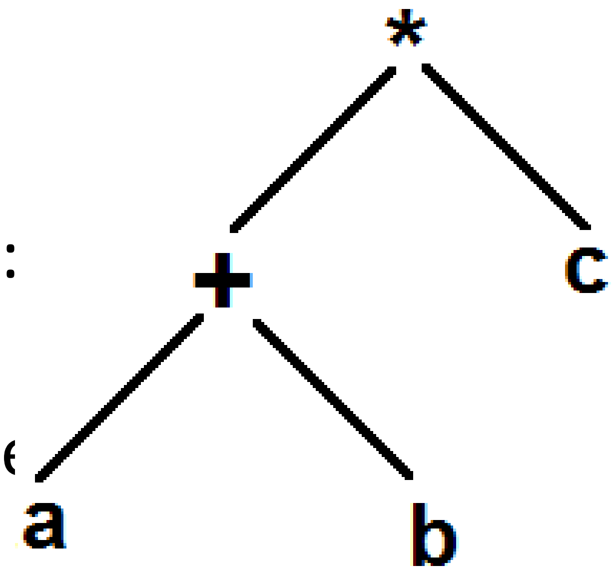
((a,b)+, c)*

3. Прямая польская запись:

*+abc

4. Обратная польская запись:

ab+c*

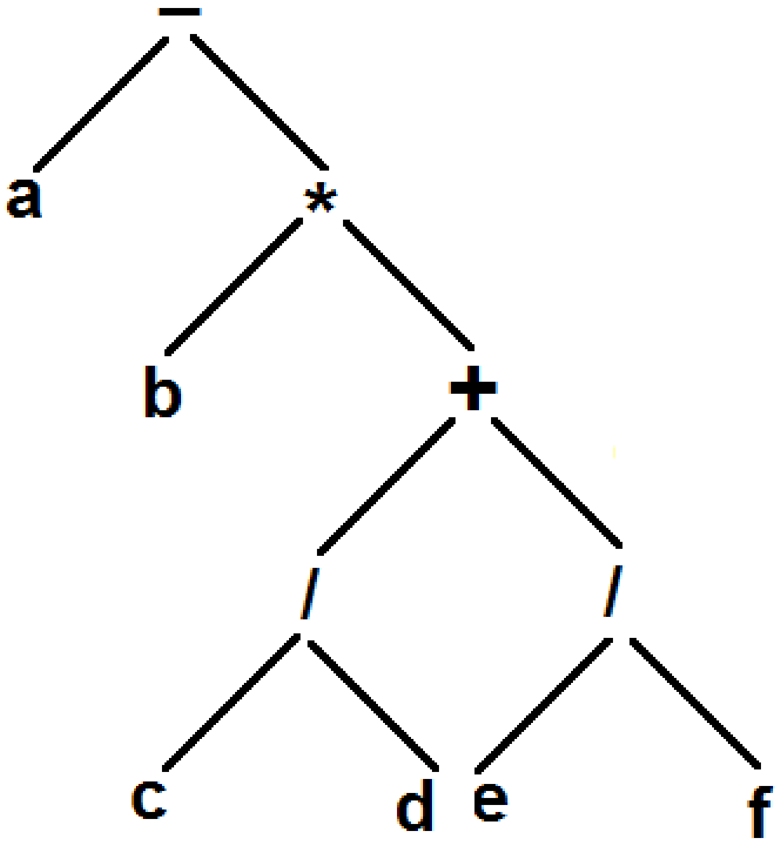


Графическое представление структуры выражения

$a - b * (c / d + e / f)$

-(a, *(b, +(/(c,d), /(e,f))))

(a, (b, ((c,d)/, (e,f)/)+)*)-



Понятие о вычислительной сложности алгоритмов

Порядок роста необходимых для решения задачи времени и памяти при увеличении входных данных

- Сложность алгоритмов оценивают по времени выполнения или по используемой памяти.
- Временная сложность
- Емкостная сложность

Вычислительная сложность алгоритма

- Обычно выражают через символ «О большое» - порядок величины вычислительной сложности

- Если алгоритм обрабатывает входы размера n за время $c*n^2$, где c - некоторая постоянная, то говорят, что временная сложность этого алгоритма есть $\mathbf{O(n^2)}$

Пример 1. Сумма элементов массива

```
for(i=0; i<n; i++)
    { //  тело цикла}
```

Пример 2. Сумма элементов квадратной матрицы

```
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        { //  тело цикла}
```

Алгоритм	Временная сложность	Пример
A1	$\log_2 n$	Поиск по бинарному дереву
A2	n	Поиск наибольшего элемента в массиве
A3	$n*\log_2 n$	Оптимальная сортировка
A4	n^2 n^3	Алгоритм сортировки методом пузырька. Умножение матриц
A5	2^n	Задачи комбинаторики

- Здесь временная сложность — это число единиц времени, требуемого для обработки входа размера **n**.
- Тогда если алгоритм **A2** может обработать за одну секунду вход размера 1000, в то время как **A5** — вход размера не более 9