Жадные алгоритмы

Айрат Хасьянов - Алгоритмы 11

Динамическое проггграммирование

- Метод решения задач с перекрывающимися подзадачами, обычно через рекуррентные соотношения;
- Можно решать задачи оптимизации, для которых выполняется **принцип оптимальности;**
- Алгоритм Флойда-Воршалла, построения оптимального бинарного дерева, решение задачи о рюкзаке;
- Используем **сохранение значений.**

Задача о размене

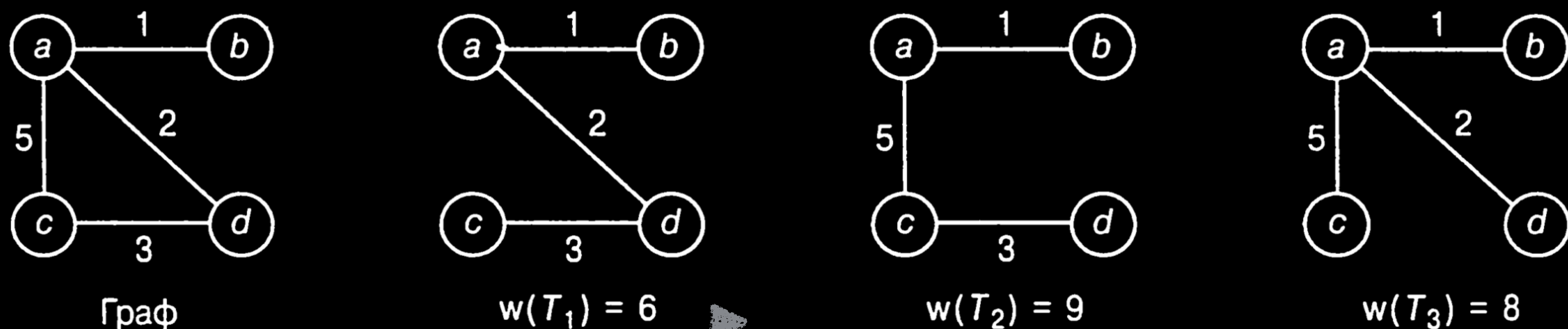
- На каждом шаге получаем частичное решение;
- Выбор на каждом шаге тчо:
  - является лучшим возможным на данном шаге;
  - Сделанный выбор не меняем

Это жаааадный алгоритм!

Задача

- Построить минимальное **остовое дерево**!
- **Остовное дерево** связанного графа - связанный ациклический подграф, содержащий все вершины графа.
- **Минимальное остовное дерево** взвешенного связанного графа - **остовное дерево** с наименьшим весом (суммой всех ребер)

Пример



Минимальное остовное дерево

Алгоритм Прима

1. Произвольно выбираем первую вершину (почему?) $V_T=\{v_0\}$

2. На каждом шаге переносим ближайшую вершину u* из $V\text{-}V_T$ (соединяется с построенным поддеревом минимальным ребром) в V_T

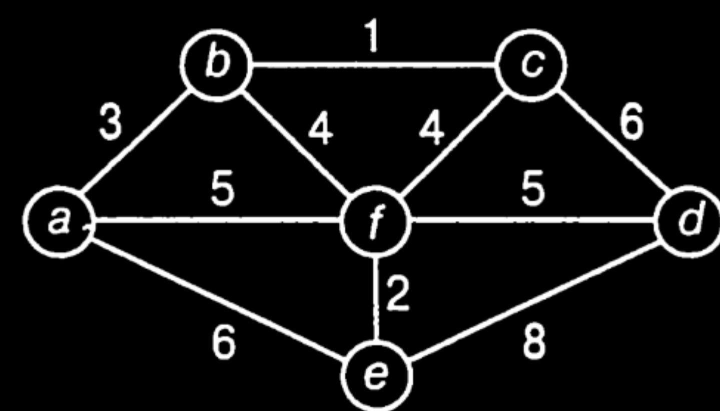
3. Для каждой вершины из $V\text{-}V_T$, соединённой с u*, обновляем метки смежной вершины, и метки расстояния до V_T

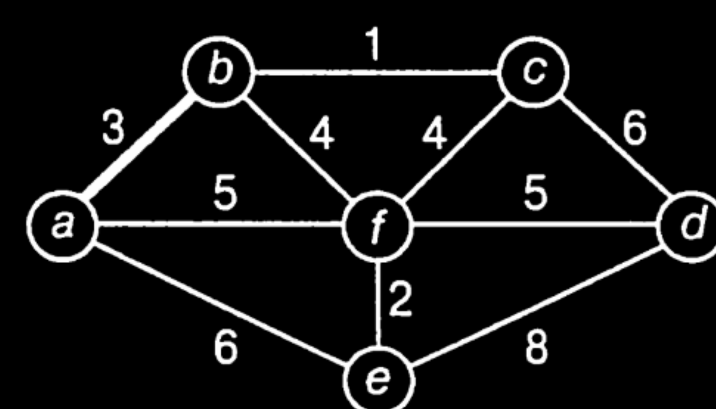
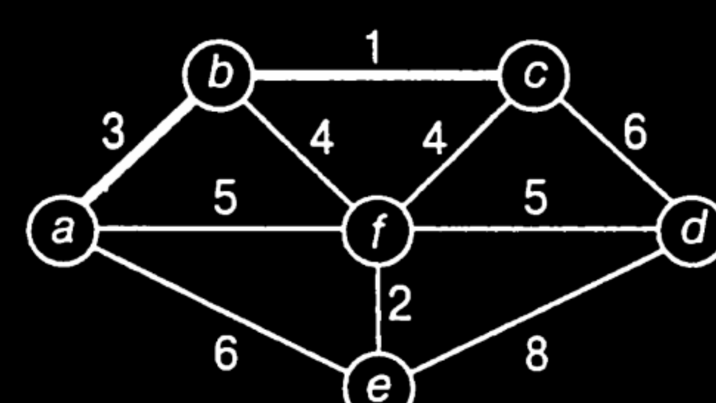
   - К каждой вершине присоединяем 2 метки: имя ближайшей вершины и вес кратчайшего ребра до этой вершины.

4. Сохраняем вновь добавленное ребро.

5. Неоднозначную ситуацию разрешаем произвольно.

Пример

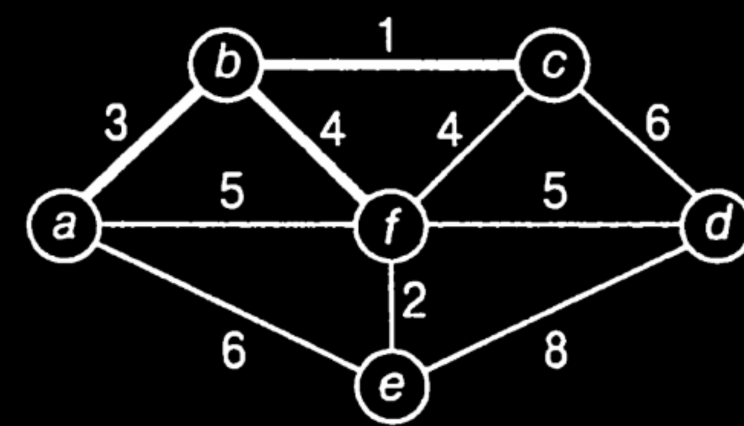


Вершины дерева	Остальные вершины*	Рисунок
a(–,–)	**b(a,3)** c(–,∞) d(–,∞) e(a,6) f(a,5)	
b(a,3)	**c(b,1)** d(–,∞) e(a,6) f(b,4)	

Продолжение

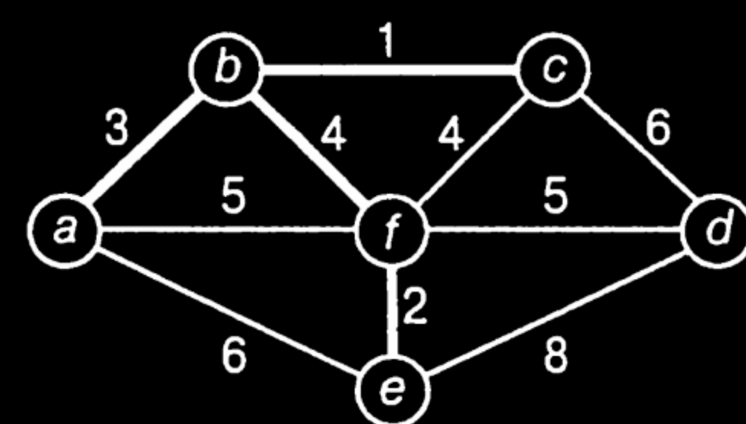
c(b,1)

d(c,6) e(a,6) **f(b,4)**



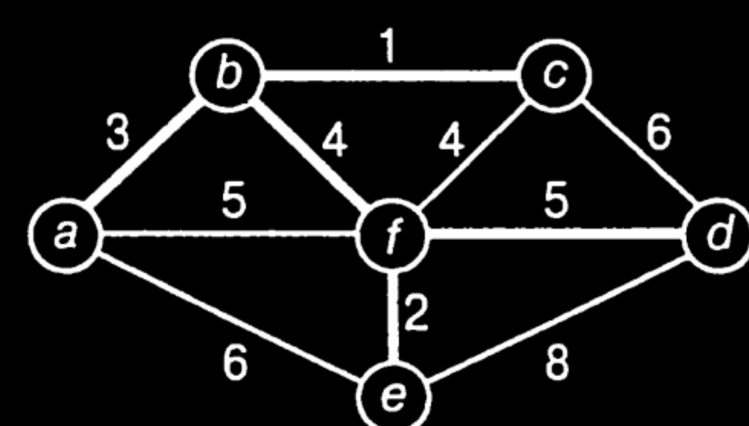
f(b,4)

d(f,5) **e(f,2)**



e(f,2)

d(f,5)



Задачи

1. По индукции доказать корректность алгоритма Прима

2. Реализовать алгоритм Прима

3. Оценить и доказать сложность (время) вашего алгоритма

Идея доказательства

1. T_0 очевидно часть минимального остова;

2. Допустим T_{i-1} тоже часть минимального остовного дерева T;

3. Пусть e_i = (u,v) - ребро минимального веса от вершины в T_{i-1} до вершины не из T_{i-1};

4. Если T_i не принадлежит ни одному минимальному островному дереву, включая T, то добавление e_i в T создаст цикл;

5. Кроме ei цикл будет содержать ребро (v',u'), где v' в T_{i-1}, а u' - не в T_{i-1}.

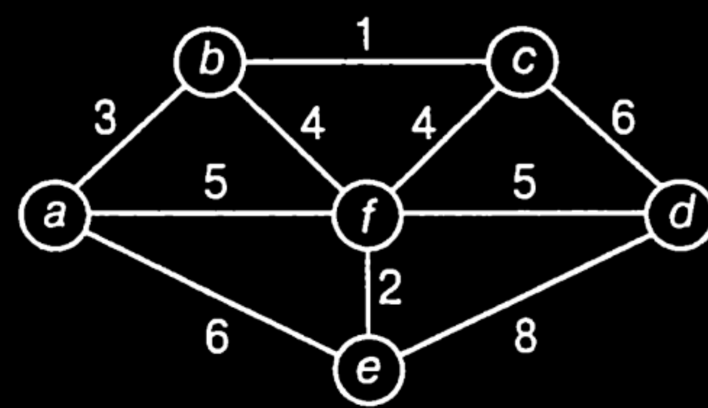
6. Уберём из T ребро e_i, получим новое остовное дерево всего графа, чей вес не будет больше веса T, так как вес e_i<=веса (v',u');

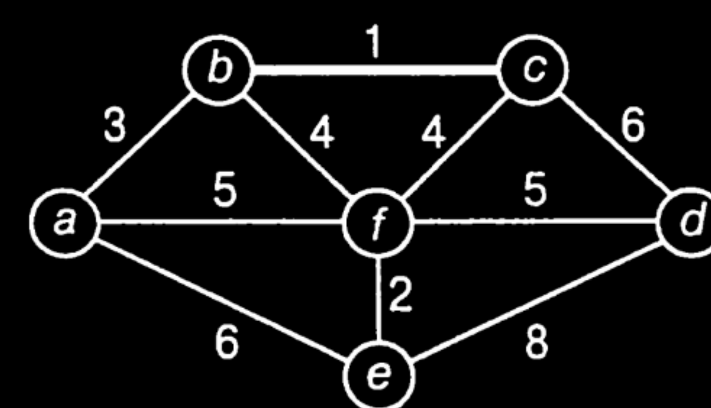
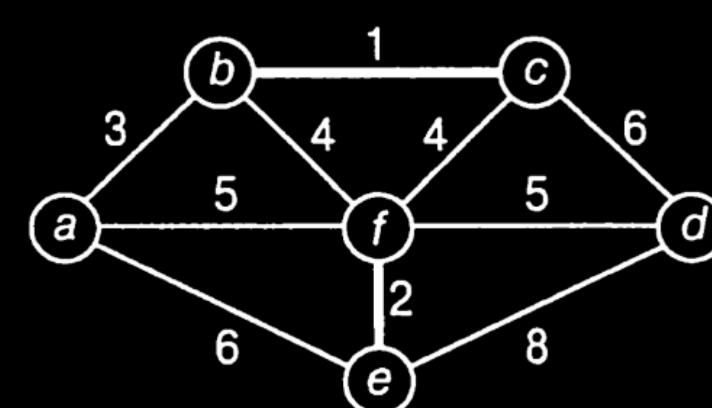
7. Новое дерево будет минимальным остовым, содержащим T_i - это входит в противоречие с предположением 4, следовательно оно не верно.

Алгоритм Крускала

- Отсортируем рёбра графа в неунывающем порядке их весов;
- Инициализируем минимальное отсевной дерево пустым подграфом;
- Очередное в упорядоченном массиве ребро добавляется в минимальный остов, если при этом не создаётся цикл, иначе ребро пропускаем.

Пример

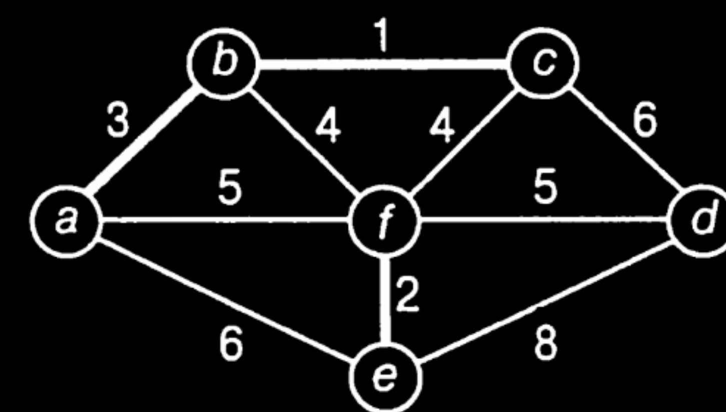


Вершины дерева	Отсортированный список ребер*	Рисунок
	bc ef ad bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ad bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	

Продолжение

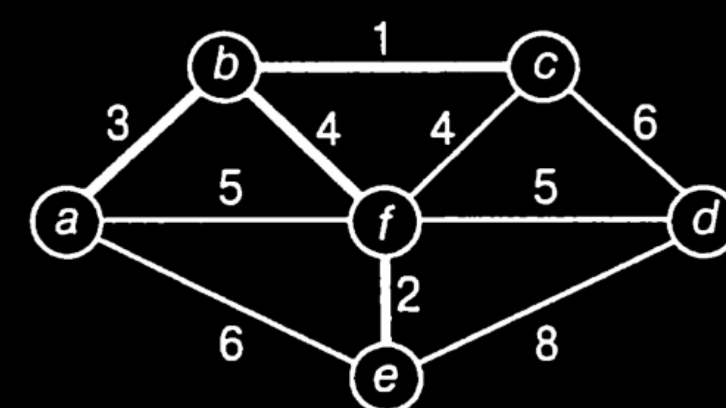
ef
2

bc	ef	**ad**	bf	cf	af	df	ae	cd	de
1	2	3	4	4	5	5	6	6	8



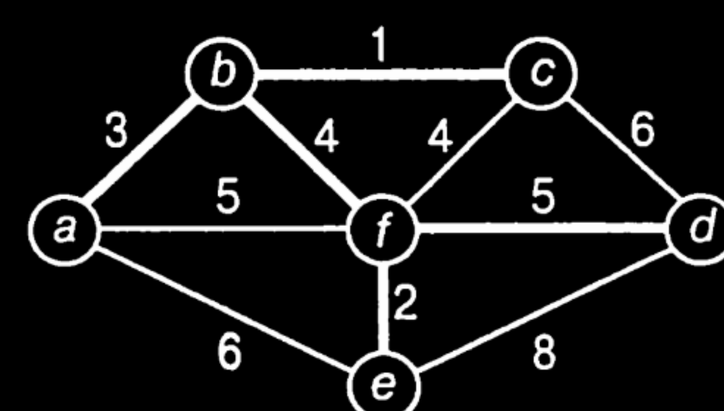
ab
3

bc	ef	ad	**bf**	cf	af	df	ae	cd	de
1	2	3	4	4	5	5	6	6	8



bf
4

bc	ef	ad	bf	cf	af	**df**	ae	cd	de
1	2	3	4	4	5	5	6	6	8



Альтернативная интерпретация алгоритма Крускала

- Начинаем с леса **|V|** тривиальных деревьев;
- Выбираем очередное (в отсортированном списке) ребро **(u,v)**;
- Находим деревья, содержащие **u** и **v**, если это разные деревья, добавляем это ребро и объединяем их в одно дерево;
- В итоге получится минимальное остовное дерево, соединяющее все вершины.

Пример

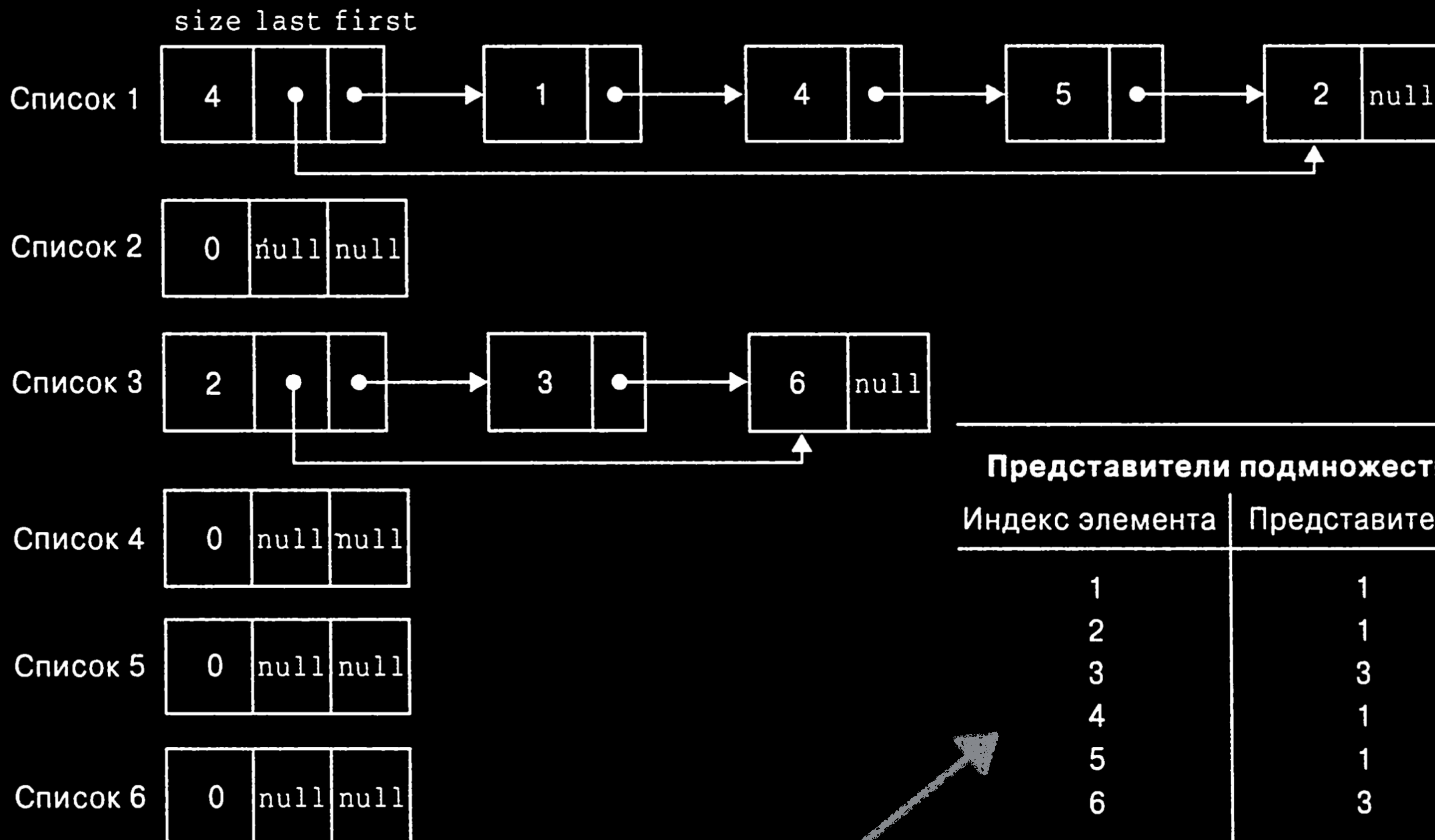
- Пусть S={1,2,3,4,5,6}

1. Исходный лес: {1}, {2}, {3}, {4}, {5}, {6}
2. Объединяем ребрами (1, 4) и (5,2): {1,4}, {5,2}, {3}, {6}
3. Объединяем ребром (4,5): {1,4,5,2}, {3,6}
4. Примечание: Веса не рассматриваем для простоты

Алгоритмы поиска и объединения

- Быстрый поиск элемента;
- Быстрое обьединение;

Быстрый поиск

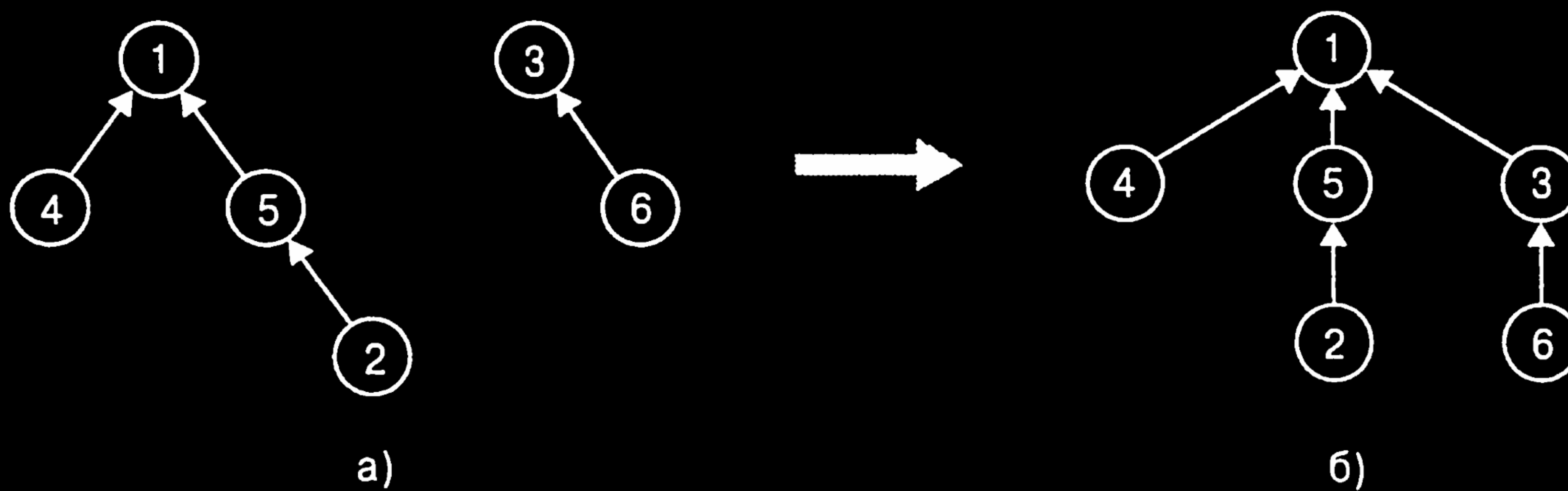


Представители подмножеств

Индекс элемента	Представитель
1	1
2	1
3	3
4	1
5	1
6	3

Используем для быстрого поиска

Быстрое объединение



а) представления подмножеств {1,4,5,2} и {3,6} в виде леса

b) результат объединения ребром (5,6)

Домашнее задание

1. Реализовать алгоритм Крускала
2. Оценить сложность алгоритма Крускала
3. Доказать корректность алгоритма Крускала (по аналогии с алгоритмом Прима)