Минимальные остовные деревья. Алгоритмы Крускала и Прима.

Работу выполнил
Студент группы
И-2-10
Бекиров Алим

Минимальные остовные деревья

Задача: в процессе разработке электронных схем необходимо получит такую компоновку из n-1 проводов, которая использует минимальное количество провода.

Дан некий связный неориентированный граф G=(V,E), где V- множество контактов, E- множество возможных соединений между парами контактов, и для каждого ребра (u,v)ЭE задан вес w(u,v), определяющий стоимость (количество необходимого провода)соединения u и v.

Минимальные остовные деревья

Необходимо найти ациклическое подмножество T , которое соединяет все вершины и чей общий вес минимален.

$$w(T) = \sum_{(u,v) \in T} w(u, v)$$

Остовным деревом(spanning tree)- называется множество T, которое ациклично и связывает все вершины, при этом данное дерево имеет минимальный вес.

Задачей поиска минимального остовного дерева (minimum spanning tree problem) называют задачей поиска дерева T.

Минимальные остовные деревья

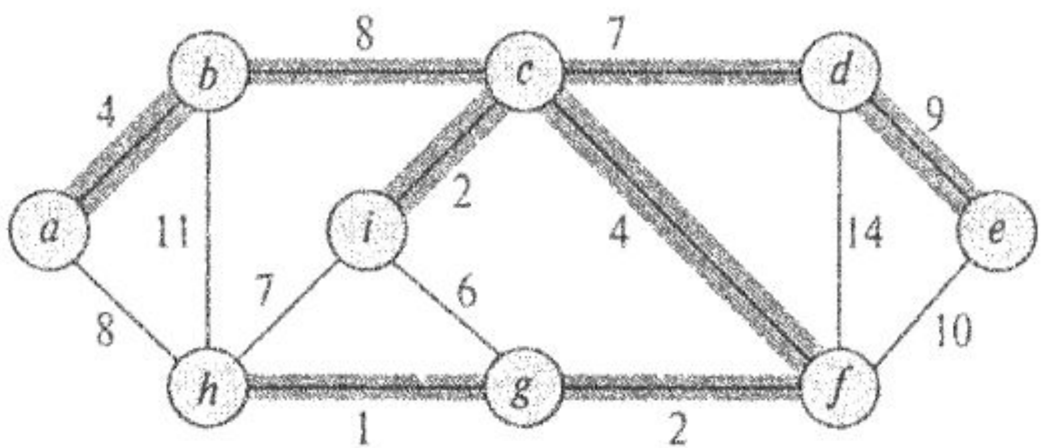


Рис. Минимальное остовное дерево связного графа

Алгоритм Крускала

- Алгоритм Крускала предназначен для нахождения безопасного ребра для добавления в растущий лес путем поиска ребра (u, v) с минимальным весом среди всех ребер, соединяющий два дерева в лесу.
- Данный алгоритм является жадным, поскольку на каждом шаге он добавляет к лесу ребро с минимально возможным весом.
- Время работы алгоритма $O(E \lg V)$.

Алгоритм Крускала

MST_KRUSKAL(G, w)

$A \leftarrow \emptyset$

for (Для) каждой вершины $v \in V[G]$

  do MAKE_SET(v)

Сортируем ребра из E в неубывающем порядке их весов w

for (Для) каждого $(u, v) \in E$ (в порядке возрастания веса)

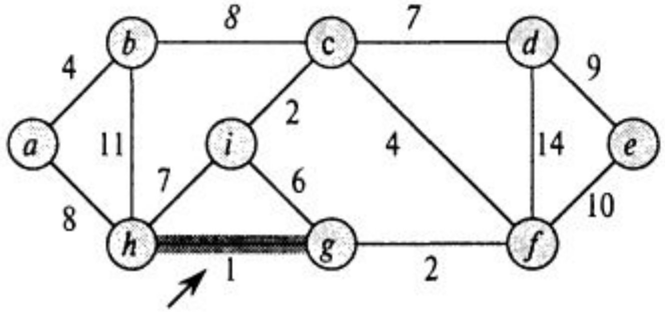
  do if FIND_SET$(u) \neq$ FIND_SET(v)

    then $A \leftarrow A \cup \{(u, v)\}$

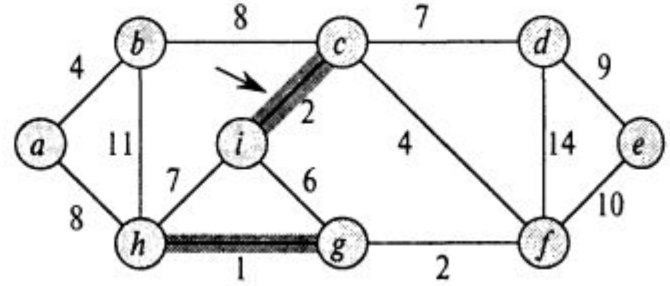
      UNION(u, v)

return A

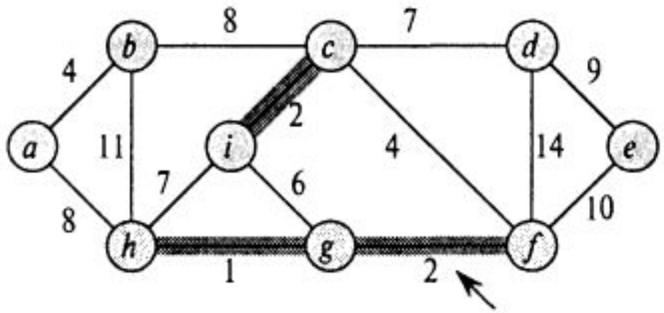
а)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

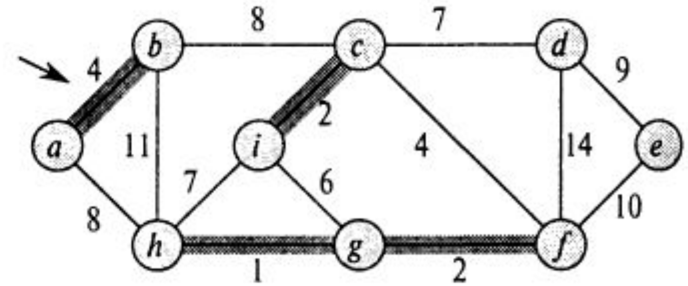
б)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

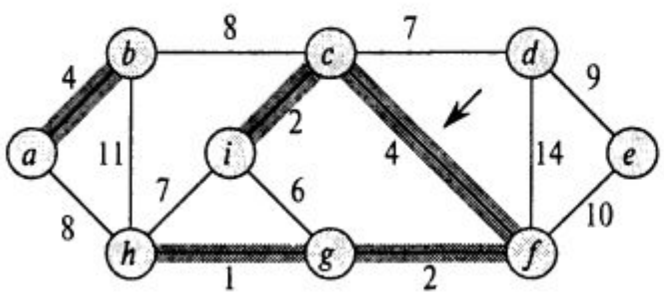
в)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

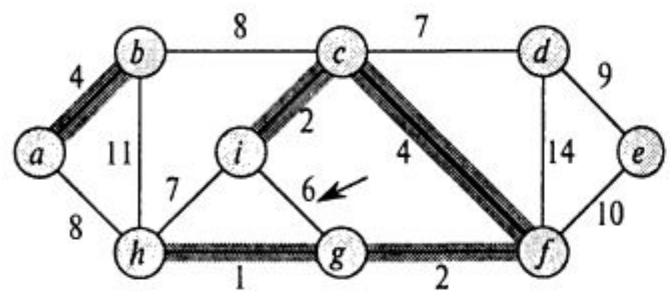
г)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

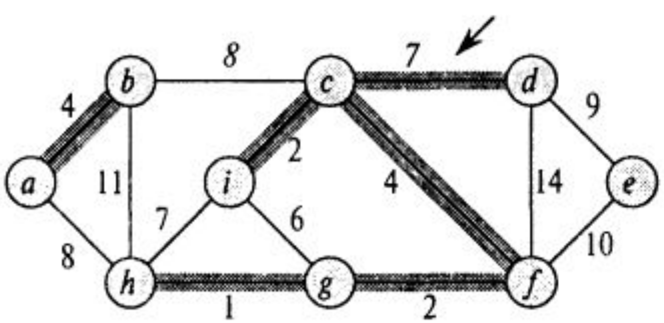
д)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

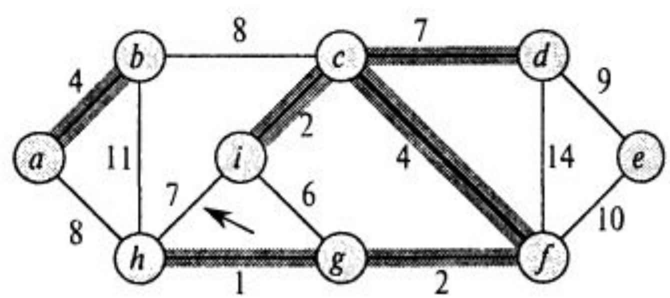
е)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

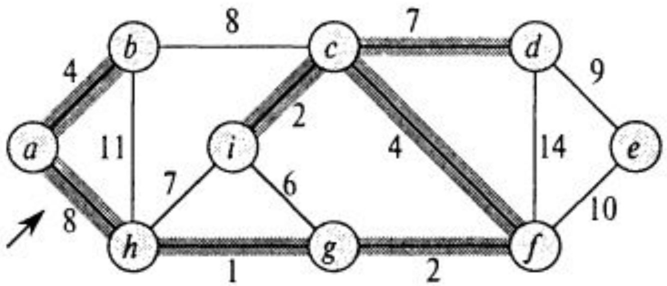
ж)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

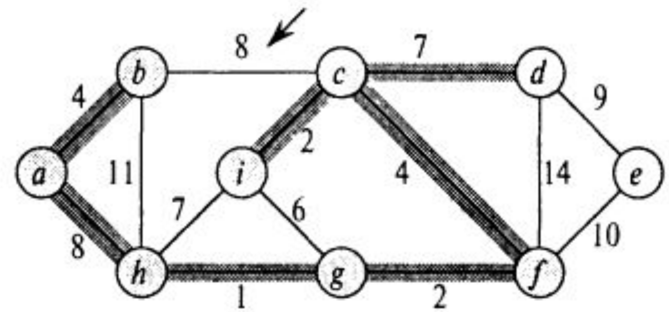
з)


b
c
d
a
i
e
h
g
f
4
8
7
9
11
2
4
14
7
6
10
8
1
2

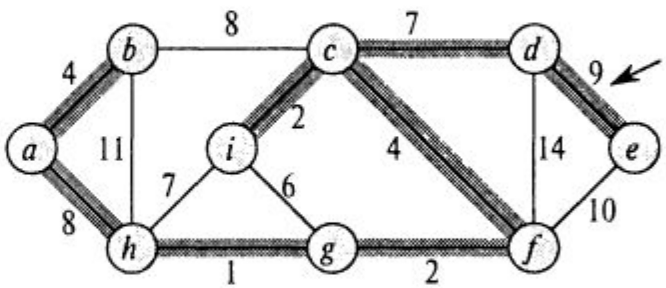
и)


b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

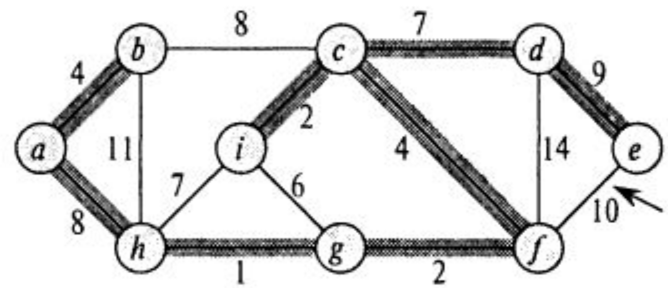
к)


b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

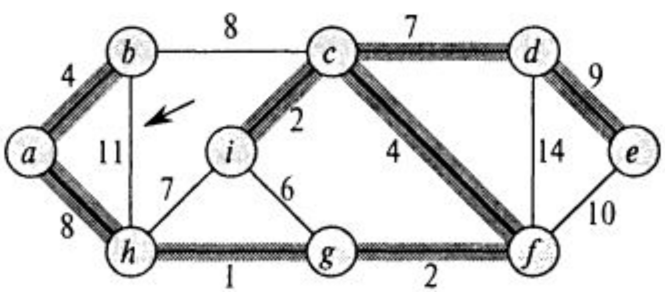
л)


b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

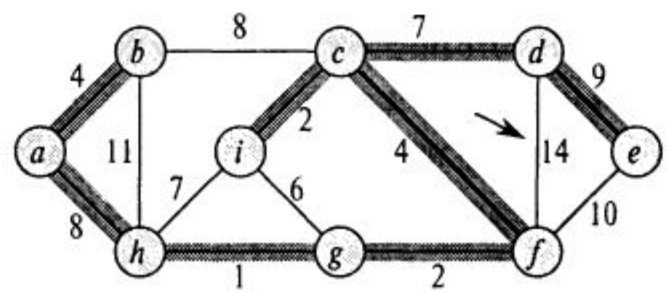
м)


b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

н)


b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

о)

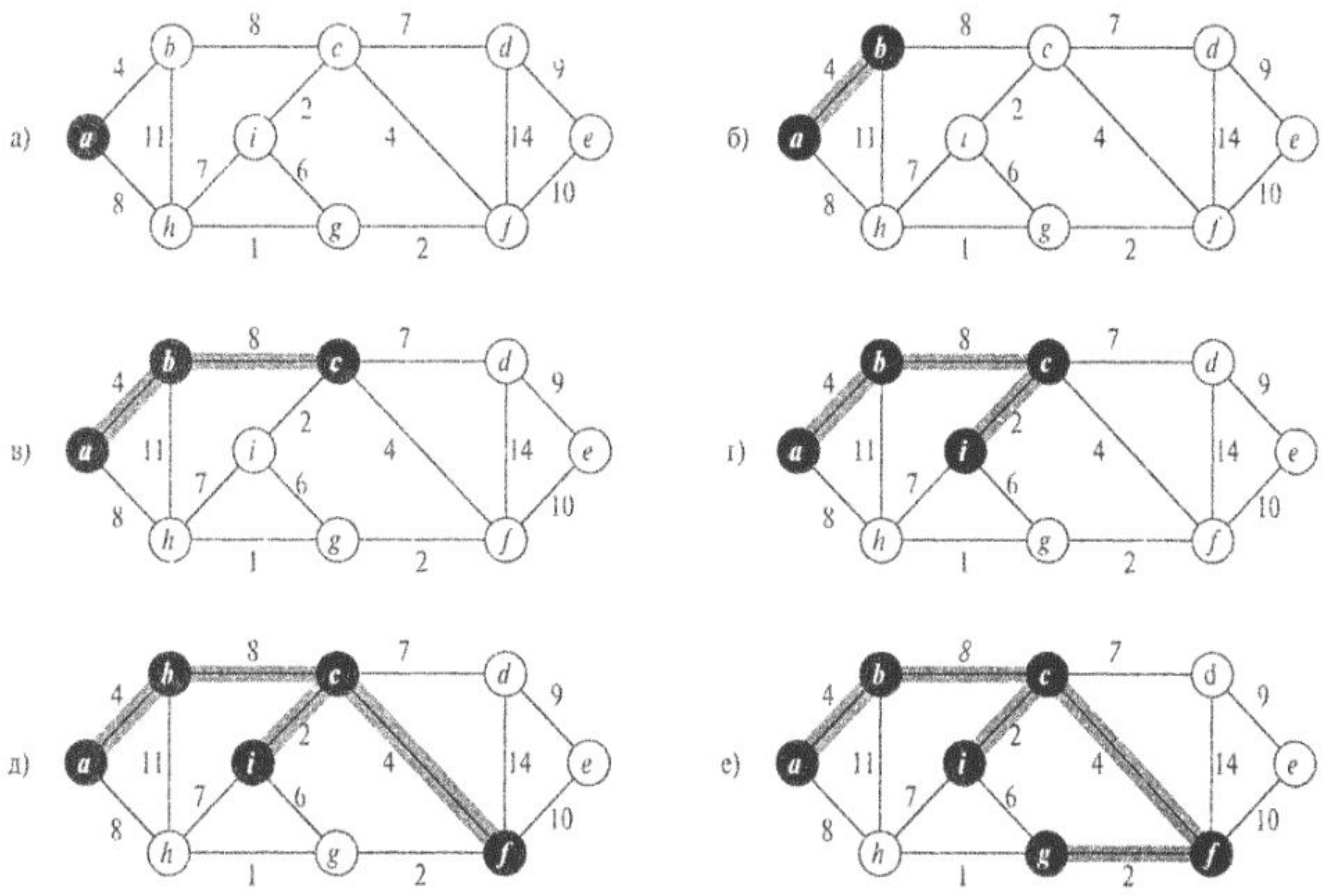

b
c
d
a
i
e
h
g
f
8
7
4
9
2
11
4
14
7
6
8
10
1
2

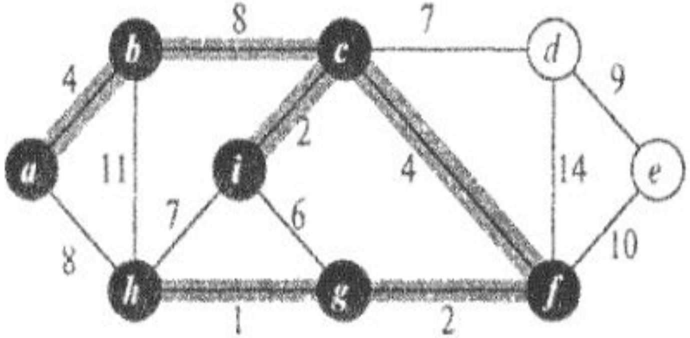
Алгоритм Прима

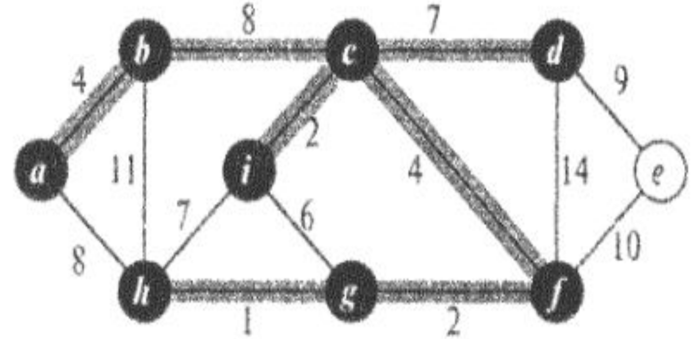
- Аналогично алгоритму Крускала. Построение дерева начинается с произвольной корневой вершиной r и растет до тех пор пока не охватит все вершины в V. На каждом шаге к дереву A добавляется легкое ребро, соединяющие дерево и отдельную вершину из оставшейся части графа.
- Время работы алгоритма $O(E + V \lg V)$.

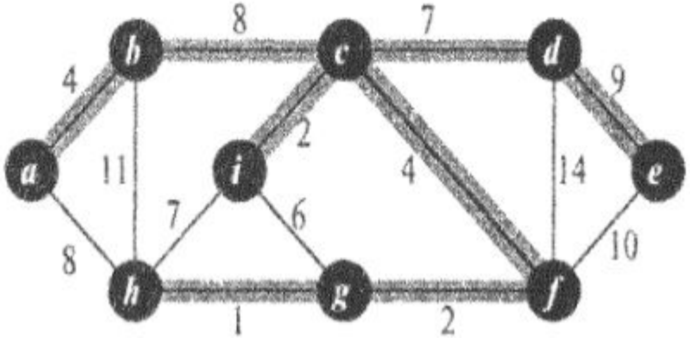
Алгоритм Прима

```
MST_PRIM(G, w, r)
for (Для) каждой вершины u ∈ V[G]
    do key[u] ← ∞
       π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT_MIN(Q)
       for (Для) каждой вершины v ∈ Adj[u]
           do if v ∈ Q и w(u, v) < key[v]
                 then π[v] ← u
                      key[v] ← w(u, v)
```


а)
б)
в)
г)
д)
е)

ж) 
a b c d e f g h i
8 7 4 9 11 2 14 4 7 6 10 8 1 2

з) 
a b c d e f g h i
8 7 4 9 11 2 14 4 7 6 10 8 1 2

и) 
a b c d e f g h i
8 7 4 9 11 2 14 4 7 6 10 8 1 2

Спасибо за внимание!