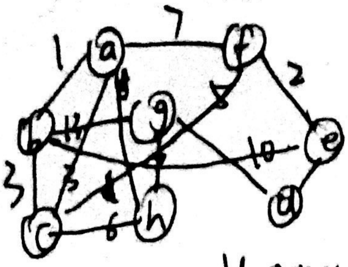
queue 的声明

```
Class Queue{
  private node front
  private node rear
  private int size
  public Queue(){
    front = null
    rear = null
    size = 0
  }
public void enqueue(int data){      // 入队
  Node newNode = new Node data
  if (rear==null){
      front = rear = newNode
  } else{ rear.next = newNode
      rear = newNode;}

  size++;
}
public int dequeue(){
  int data = front.data;
  front = front.next;
  if (front == null){
     rear = null;
  }
  size--;
  return data;
}
```

完全二叉树 左偏树 都可左偏.



P3A表.

{a,b}最短所以选择{a,c} → {a,b,c} → {a,b,c,f}

← {a,b,c,f,...}

Vertex v	best v and weight	选{c}	{c,f}	
a	n/a	n/a	n/a	n/a
b	n/a	n/a	n/a	n/a
c	{c,a},3	n/a	n/a	n/a
d	nil,∞	nil,∞	nil,∞	[illegible]
e	{e,b},10	{e,b},10	{e,f},2	n/a
f	{a,f},7	{c,f},5	n/a	n/a
g	{g,h},13	{g,h},13	{g,h},13	{g,h},13
h	{a,h},8	{c,h},6	{c,h},6	{c,h},6

SCC(强连通分量) 对 u和v 存在双向路径.

寻找SCC, Tarjan算法. 1.使用DFS遍历图,同时用栈来存储访问的节点。2.为每个节点分配一个唯一的索引。当发现一个强连通分量时,将栈中的节点出栈,直到找到当前节点,并将这些节点标记为强连通分量。时间复杂度 O(V+E)

单源最短路径 - Dijkstra算法

1. 为每顶点设一个初始距离值。起始点距离为0,其它所有点距离为无穷大,创建空集合,用于存储已经确定最短路径的顶点。
2. 对于当前顶点的每个邻接点,如果通过当前顶点到达邻接点距离更短,标为当前顶点,将当前顶点加入到已处理的集合。重复所有,直到所有点都被加入已处理的集合。

负权不可用. → $O(V^2)$ 数组 　 $O((V+E)\log V)$ 优先队列(堆)

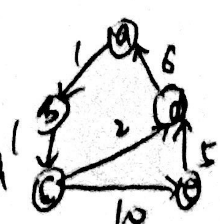
Prim算法 将图转化为最小生成树(MST)

从图上任意顶点开始,将其标记为已访问。将所有已访问的顶点中选择最小的边加入MST中,然后标记顶点为已访问的边,重复直到所有顶点被访问,每一步都选最小.

V是顶点数, E表示边数.

O(n)=邻接矩阵$O(V^2)$,优先队列$O(E\log V)$.

邻接表 O(V+E)



V	dist(v)	parent v
a	∞	|
b	∞	|
c	0	|
d	∞	|
e	∞	

每次选择一个节点在图里删掉

如果有更短的最短路径发生替换、从c开始.

remove → c

V	dist(v)	parent v
a	∞	null
b	∞	null
c	0	null
d	2	C
e	10	C

←

V	dist	parent
a	∞	nil
b	0	nil
c	2	c
d	10	c
e		

←

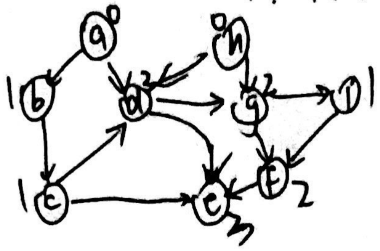
V	dist	parent
a	9	a
b	0	nil
d	2	C
e	10	C

⇐ c → d, e; a; b

directed graph
{V, E} ⇒ V代表所有节点，E代表边（包括有向边的集合）→所有边均有向.
cycle: 环 {V_1, V_2, V_3, V_4, V_1}

Heap 排序，每次用 小根堆，连放最后一个上来，然后与左右小的 swap.

→有向无环图.
Topological Sort 从上往下，从入度最小的开始的删除，删除后入度减去.



a h b c d g i f e
h a b c d g i f e 都可以.

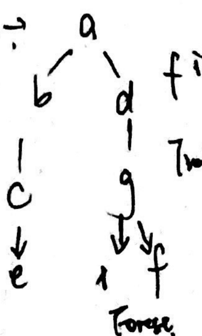
BBST O(n) space
O(log n) per predecessor query, Insert, delete

DFS
dfs{
for{
if(!visited){dfs()}
}
}

bfs{
while(!queue.isEmpty()){
sout(X)
for( ){
if(!visited.contains(x)){
visited.add(X)
queue.add(X)
}
}
}.

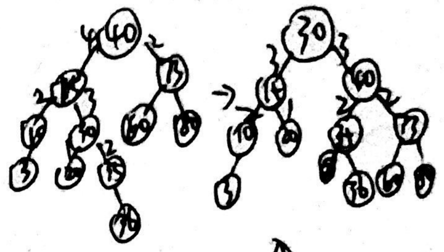
dfs 反向后

SSSP BFS:



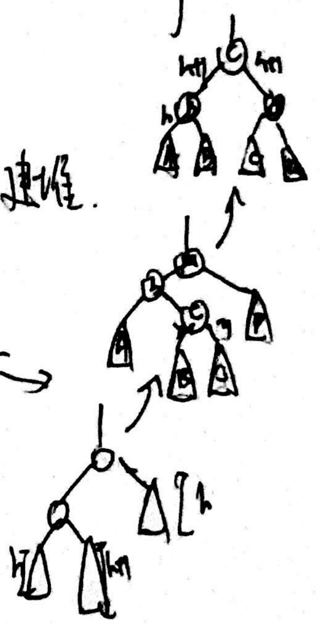
f is not reachable from a
T=O(V+E)

例 AVL

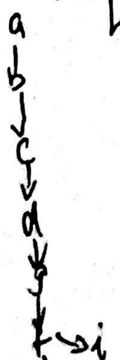


Root-fix O(n) 建堆.

三个点



SSSP DFS



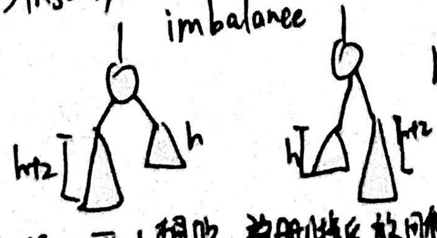
Forest h.

repeat until S is empty
Let v be the first
Does v has a available neighbor.
yes: → let be u
no: pop v from S, color v → unavailable.
If there are still available create a new tree.

Search Pattern
a a b a a a b b

priority Queue (heap)
O(n) space
O(log n) → insert, deletemin time

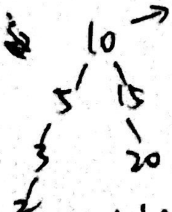
imbalance



→改三个指针

left left case

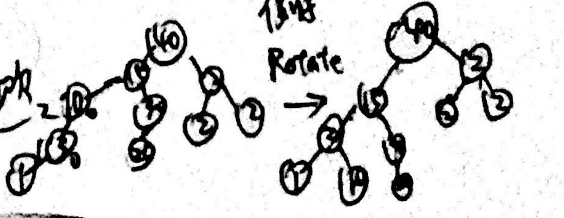
bbst 10 → 不误，左右差最多1.



Huffman 两个相加，放回排序放回.
左0右1
证明 $L = \sum_{i=1}^{n} f_i d_i$
最小的 f_i 与 l_i 相乘加起来最小

Recall x=h或h+1

Rotate



preorder: root, left, right
Postorder: left, right, root
Inorder: left, root, right
Level-order: top to bottom, left to right