λ	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞

Father	1	2	3	4	5	6

T	1	2	3	4	5	6
	Y	Y	Y	Y	Y	Y

distance between current node and adjacent to current node

u - current node w - adjacent to current node

p - is father of adj node, therefore <= u

λ[w] - is taken from λ table based on adj w

once visited 1 -> 3 we do not have to do 3 -> 1

T - current node u, if visited X

if (λ[w] ≥ λ[u] + d(u,w))

λ[w] = λ[w] + d(u,w)

P[w] = u

Step 1: u = 1; w = 2

λ[w] = ∞ ≥ λ[1] + d(u,w)

∞ > 0 + 3; λ[2] = 3; P[2] = 1

w = 4; λ[4] = ∞ ≥ 0 + 2 ✓; λ[4] = 2; P[4] = 1

w = 3; λ[3] = ∞ ≥ 0 + 4 ✓; λ[3] = 4; P[3] = 1

Step 2: u = 4; w = 3

λ[3] = 4 ≥ 2 + 1 ✓; λ[3] = 3; P[3] = 4

override previous

Step 3: u = 3; w = 2; 3 ≥ 3 + 1 ✗

w = 6; ∞ ≥ 3 + 3; λ[6] = 6; P[6] = 3

Step 4: u = 2; w = 5; ∞ ≥ 3 + 2; λ[5] = 5; P[5] = 2

Step 5: u = 5; w = 6; 6 ≥ 5 + 4 ✗

T	1	2	3	4	5	6
	X	X	X	X	X	X

Shortest Path, last visually

From 1 to 6 = 1 -> 4 -> 3 -> 6; 1 to 5 = 1 -> 2 -> 5; 1 to 4 = 1 -> 4; 1 to 3 = 1 -> 4 -> 3; 1 to 2 = 1 -> 2

once false λ, F, T stay the same

λ	1	2	3	4	5	6
	0	3	3	2	5	6

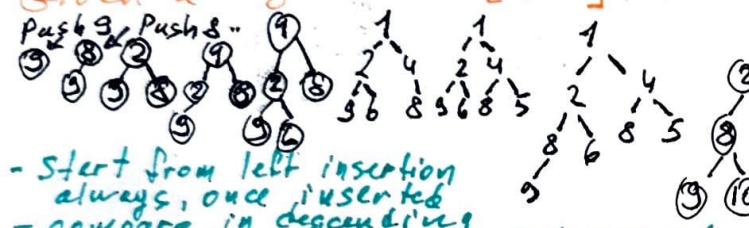
F	1	2	3	4	5	6
		1	4	1	2	3

override

T	1	2	3	4	5	6
	X	X	X	X	X	X

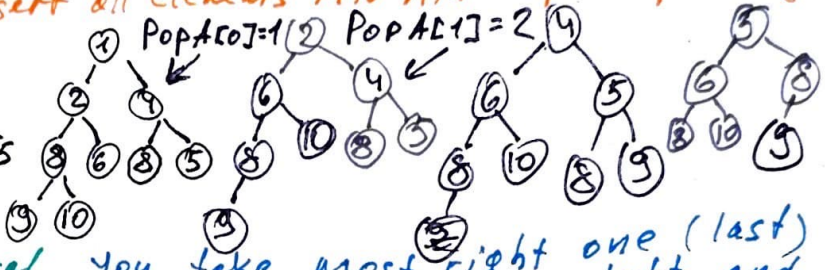
Given a sequence A = [9, 8, ..] insert all elements into Min-Heap for priority queue

Push 9, Push 8, ..



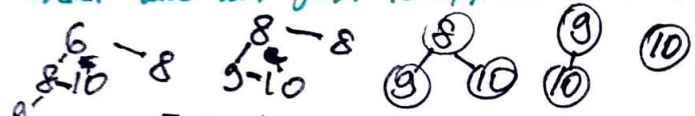
Pop A[0] = 1

Pop A[1] = 2



- start from left insertion always, once inserted
- compare in descending order and not just level, but several

you take most right one (last)
- put on top, compare left and right (always two sides for smallest) and go that branch.
- it is not just one node comparison, it compare until it's in right position



A = {1, 2, 4, 5, 6, 8, 8, 9, 10}

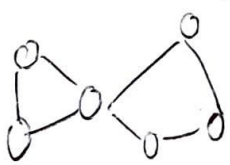
c) Explain how do we get order n log n first for insert, then delete

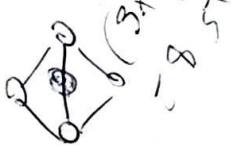
The height of a tree is O(log2 n). When inserting elements using min-heap, the worse case time complexity would be the worse case at a tree (log2 n). Therefore, if we have n elements to insert or delete, the time complexity becomes O(n log2 n)

```
int main()
{ int A[] = {10, 5, ..., 4}
  MinHeap M(10);
for (int i=0; i<4; i++)
M.Insert(A[i])
for (int j=0; j<4; j++)
    M.Delete(A[j])
```

i-edges Power Set

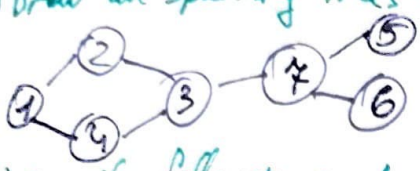
$2^4 = 16$

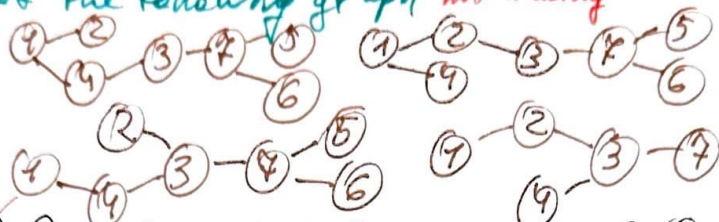




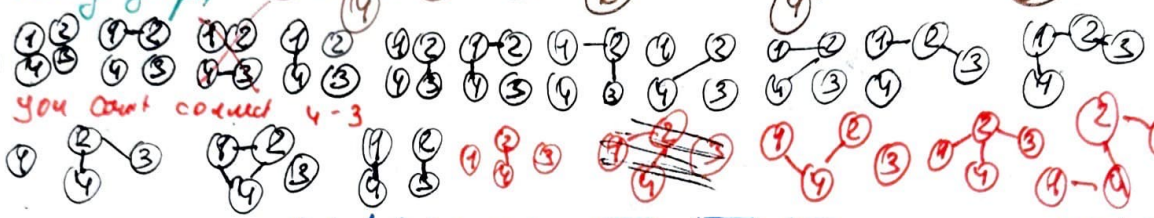
$G=(V,E)$ set of vertices, edges. Subgraph does not have to be connected. Degree
$\sum d(V) = 2e$ spanning tree has to have all vertices, connected, but no cycle. Power
set is all possible subset $2^V = 16$ walk can repeat edges, V. Trail is walk without re-
petition of edges (but allowed V) closed trail that first and last V are the same
Forrest is several trees. Path we can not repeat V. Counterpositive there is pair that is
not connected. Complete graph is everybody knows everybody n^{n-2} $4^{4-2} = 4^2 = 16$
a) Graph $G=(V,E)$ is an ordered pair, such that V is a finite set whose elements vertices
and E is a set of unordered pairs of distinct vertices of V, called edges. Spanning
subgraph of $G=(V,E)$ is subgraph of G, such that $V'=V$ (it contains all original
vertices of G) spanning tree $G=(V,E)$ is a tree V' that contains all the ver-
tices of G ($V'=V$) Trail btw vertices u and v $G=(V,E)$ is walk without repetion of edges.
Degree of vertex u of a graph $G=(V,E)$ is number of edges incident at V it

1) Draw all spanning trees of the following graph not creating new edge but removing

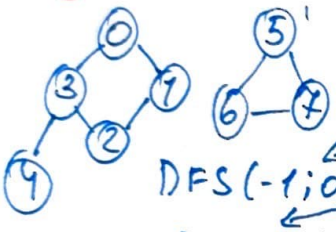




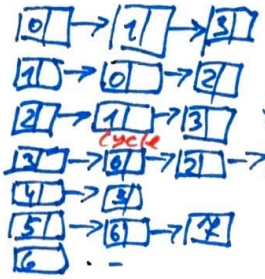
2) Consider following graph



you can connect 4-3



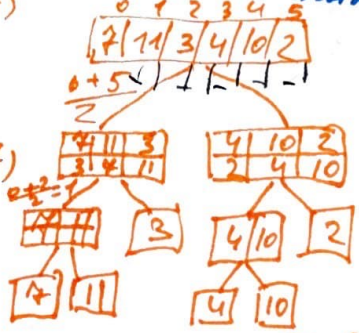
visited	0	1	2	3	4	5	6	7
	F	F	F	F	F	F	F	F
	T	T	T	T	T			



```
main()
visited[8]=false
int numpcomp=0
for(int i=0; i<8; i++)
{ if(!visited[i])
  { DFS(-1,i)
    numcomp++
  }
}
```

DFS(-1;0) → DFS(0,1) → DFS(1,2) →
→ DFS(2,3) → DFS(3,4) ;; Another main call DFS(-1;5) → ....

```
MergeSort(int first, int last)
{ if (First<Last)
  { int mid = First+last/2
    MergeSort(First, Middle)
    MergeSort(middle+1, last)
    Merge(First, Last)
  }
```

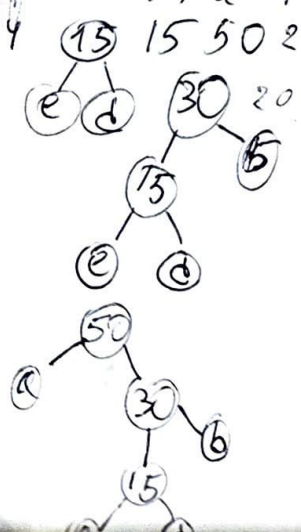


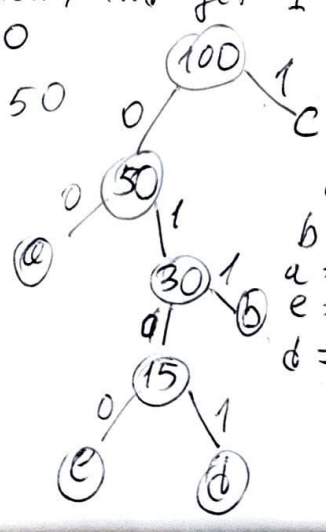
$w(n) = w(n/2) + w(n/2) + n$
$w(n) = 2w(n/2) + n$ // $n = 2^k$
$w(n) - 2w(n/2) = n$
$w(2^k) - 2w(2^{k-1}) = 2^k$
① $w(k) - 2w(k-1) = 2^k$
② $w(k+1) - 2w(k) = 2^{k+1}$ // $k \to k+1$
③ $2w(k) - 4w(k-1) = 2^{k+1}$ // ① · 2
②-③ $w(k+1) - 4w(k) + 4w(k-1) = 0$
$x^{k+1} - 4x^k + 4x^{k-1} = 0$
$x^{k-1}(x^2 - 4x + 4)$
$(x-2)(x-2) \Rightarrow x = 2$
$w(k) = c_1 \cdot 2^k + c_2 \cdot k 2^k$
$n = 2^k$ $w(n) = c_1 n + c_2 n \cdot \log_2 n$
Merge sort $n \log n$

Ex: $\lim_{n\to\infty} \frac{n}{n \log n} = 0$ $\frac{1}{100000} = 0$
$\lim_{n\to\infty} \frac{n \log n}{n} = \infty$

	a	b	c	d	e
10	20	15	50	10	5

11 take 2 smallest
12 combine them and get 1
14 15 15 50 20





c = 1 - 50
b = 011 - 45
a = 00 - 40
e = 0100 - 20
d = 0101 - 40
195