COP3503 2/22/2024

1. Huffman Coding - show code
2. Brute Force / Greedy Combo
3. Graph Introduction

Welcome Party

Input upto 300 names First/Last
Goal calc min # tents (First LI., Last LI)
so all people have a party to attend.

First 26 possible
Last 17 possible } 2^{17} possible subsets of tents we can do via last name.

If I know my last name tents are C, M, N

Name
A. A.
~~A. C~~
~~A. M~~
B. R
~~B. N~~
~~D. M~~
~~F. M~~
~~Q. N.~~
~~R. C.~~

1. Cross off everyone who has a party. Note: not making anymore last name groups.
2. Greedy give groups to everyone by 1st name who needs it!

3 Last + 2 First = (5)

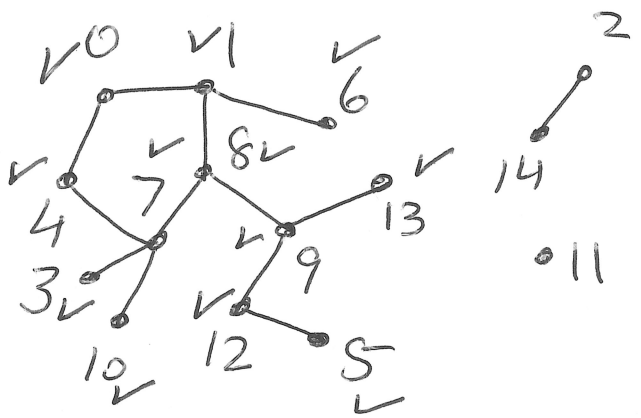
```
int ans = 17;
for (int mask = 0; mask < (1<<17); mask++) {
    // mask is which last name tests we have
    int groups = Integer.bitCount(mask); // # last name groups

    int firstmask = 0;
    for (int i = 0; i < n; i++) {
        ~~if ((1<<last[i][0] - 'A')~~
        if(( (1 << (last[i][0]-'A')) & mask ) != 0)
            continue;
        firstmask |= (1 << (first[i][0]-'A'));
    }
    ans = min(ans, groups + Integer.bitCount(
                                firstmask));
}
```

Depth First Search

Goal: Simply mark all reachable vertices from a start vertex.

Can use to mark all connected components in a graph




```
dfs(v):
    visited[v] = true;
⇒   for (int u : g[v])
        if (!visited[u])
            dfs(u)
```

~~dfs(10)~~
~~dfs(3)~~ ~~dfs(4)~~
~~dfs(7)~~
dfs(8)
~~dfs(6)~~
dfs(1)
dfs(0)

~~dfs(5)~~
~~dfs(12)~~ ~~dfs(13)~~
~~dfs(9)~~
~~dfs(8)~~
~~dfs(1)~~
~~dfs(0)~~

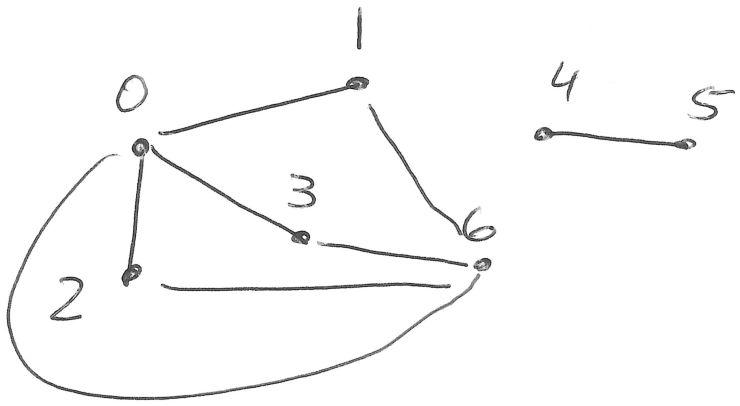
Order Visited: 0, 1, 6, 8, 7, 3, 4, 10, 9, 12, 5, 13

2, 14

11

Graphs

Graph = collection of vertices and edge.
An edge is either an unordered or ordered pair.
Draw vertices as dots + edges as lines btw dots



$|V|$ = # vertices
$|E|$ = # edge

adj matrix $|V| \times |V|$

	0	1	2	3	4	5	6
0	0	1	1	1	0	0	1
1	1	0	0	0	0	0	1
2	1	0	0	0	0	0	1
3	1	0	0	0	0	0	1
4	0	0	0	0	0	1	0
5	0	0	0	0	1	0	0
6	1	1	1	1	0	0	0

m[i][j] = 1
if edge btw
i and j
= 0 otherwise

adj list

0 → 1, 2, 3, 6
1 → 0, 6
2 → 0, 6
3 → 0, 6
4 → 5
5 → 4
6 → 0, 1, 2, 3

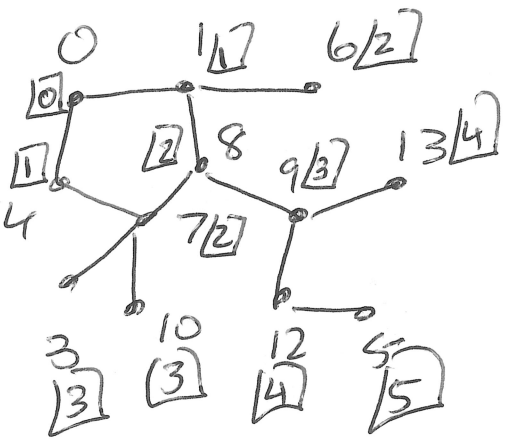
Array of ArrayList

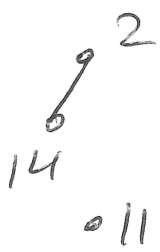
I would always use this
except when you're running
Floyd Warshall's Alg, or calc
# of diff paths of a fixed
edges btw all pairs of vertices.

Breadth First Search

DFS - recursive
does not give
shortest distances

Iterative, does provide shortest distances (# edges to traverse to get from i to j.)






```
bfs(v):
    int dist[n];
    fill(dist, ∞)
    dist[v]=0;
    queue q;
    q.offer(v);
    while (q.size()>0):
        int cur=q.poll();
        for (int next:g[cur]
            if (dist[next] !=
                ∞)
                continue;
            q.offer(next)
            dist[next] =
                dist[cur]+1;
```

Q : ~~0, 1, 4, 6, 8, 7, 9, 3, 10, 12, 13, 5~~

Dist	
0	0
1	1
4	1
6	2
8	2
7	2
9	3
3	3
10	3
12	4
13	4
5	5