② Q:n → Short discussion / Approach [Try]

Graph → Complete

Floyd warshal

Job Sequencing

$O(n\log n) + O(n * \text{no. of days}) \approx O(n^2)$

n Jobs

Job id	Deadline	Profit
A	4	300
E	4	260
D	3	250
F	2	210
~~H~~	~~2~~	~~200~~
~~B~~	~~3~~	~~150~~
G	5	150
~~C~~	~~1~~	~~100~~

—	D	—	A	—
1	2	3	4	5

(1) Profit → sort → $n\log n$

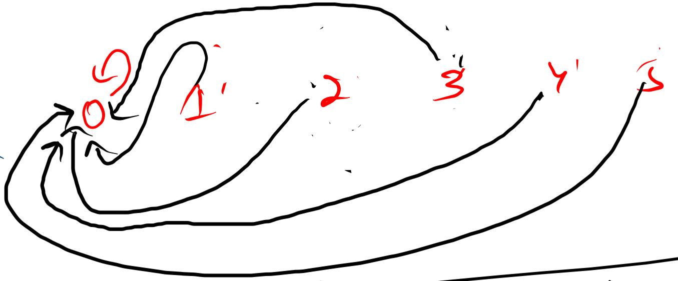
Profit ⇒ 0 + 300 + 260 + 250 + 210 + 150

Job Sequencing

F → D → E → A → G

1 2 3 4 5

DSU without union by Rank



$O(n \log n) \approx O(n \log n) + O(n \times \log(\text{no of Jobs}))$

Profit ⇒ 300 + 260 + 250 + 210 + 150 →

Find → O(1) commonly

Jobid	Deadline	Profit
A	4	300
E	4	260
D	4	250
F	3	210
M	2	200
B	3	150
h	5	180
C	1	100

Two strings `X` and `Y` are similar if we can swap two letters (in different positions) of `X`, so that it equals `Y`. Also two strings `X` and `Y` are similar if they are equal.

For example, `"tars"` and `"rats"` are similar (swapping at positions `0` and `2`), and `"rats"` and `"arts"` are similar, but `"star"` is not similar to `"tars"`, `"rats"`, or `"arts"`.

Together, these form two connected groups by similarity: `{"tars", "rats", "arts"}` and `{"star"}`. Notice that `"tars"` and `"arts"` are in the same group even though they are not similar. Formally, each group is such that a word is in the group if and only if it is similar to at least one other word in the group.

We are given a list `strs` of strings where every string in `strs` is an anagram of every other string in `strs`. How many groups are there?

t a r s
a r t s
→ ~~dis~~ x Similar

Transitive

A = B , B = C

A = C

tars ≈ rats

rats ≈ arts

→ (tars ≈ arts)

Similar $\Rightarrow$ DSU

```
Input: strs = ["tars","rats","arts","star"]
Output: 2
```

{ tars, rats, arts, star }

n -> no. of words

```
for(i=0; i<n; i++){
    for(j=i+1; j<n; j++){

    }
}
```

$O(n^2 * \text{word len}) \approx O(n^3)$

tars

rats $\rightarrow$ tars $\leftarrow$ arts

star

Similarity(w1, w2)

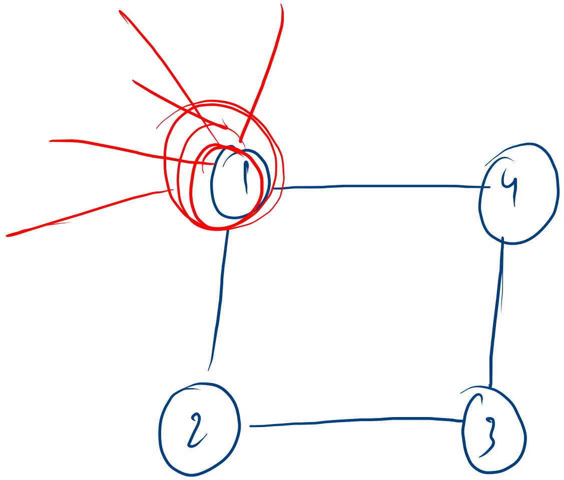
- -> w1.len == w2.len ✗
- -> w1.charType == w2.charType ✗
- -> Traverse & check { Count==2 => similar }

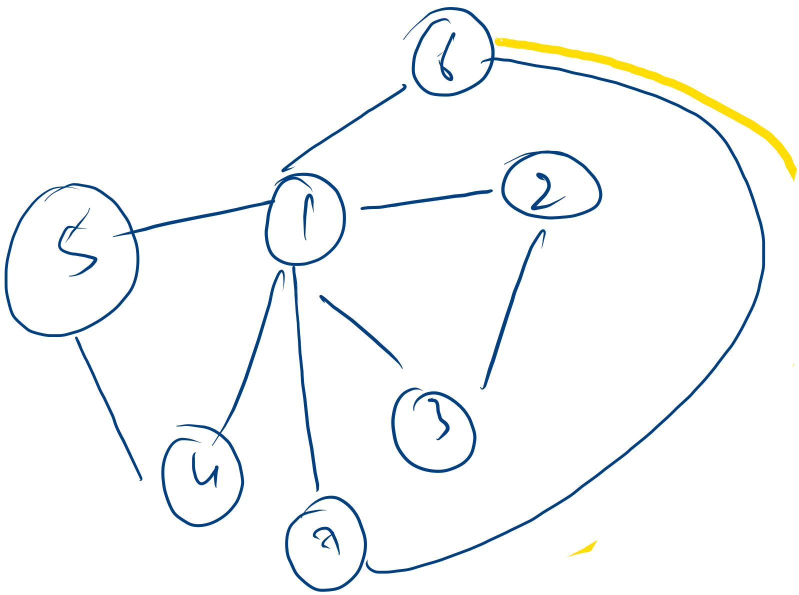
Diff. Consecutive character count

Hamiltonian → vertices

Eulerian → Edges

Graph → edges exactly once / vertices atleast once — Eulerian





0	1	2	3	4	5	6	7
10	16	12	14	15	31	5	8

99

{val, oidx}

0	1	2	3	4	5	6	7
(5,6)	(8,7)	(10,0)	(12,2)	(14,3)	(15,4)	(16,1)	(31,5)

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

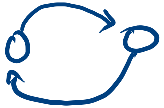
Count => 1+1+1+1+1+1+1+1 => 8

ans => 0+7

{10} ⇒ 0

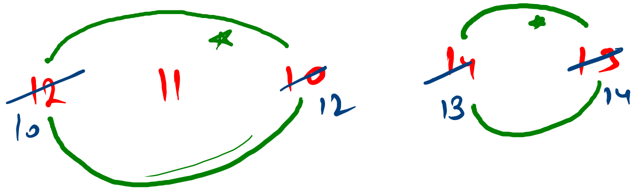
1 swap ⇒ 1 correct

Cycle ⇒ 2 → 1 swap



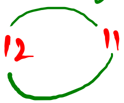
{~~11~~ ~~10~~} ⇒ 1

12 ~~14~~ 10 11 ~~12~~ 14 ⇒ 3



10 12 11 ⇒ 1

~~12~~ ~~10~~ ~~11~~ 12 10 11 ⇒ 2



17			5	
~~18~~	18		20	
~~21~~		19	~~18~~	21
~~14~~	~~20~~	~~21~~	~~17~~	~~18~~

⇒ 4

10 ~~12~~ 11 ~~10~~ 12   13 ~~14~~ ~~13~~ 14

min ⇒ 2

1 → 1 ele correct

1 ele corre

1 ele corr

1 → 2 ele correct

swap	1	1	1	1
correct	1	1	1	2

Cycle Size ⇒ 5 (n)

swaps ⇒ 3+1 ⇒ 4 $(n-1)$

Shortest path algo.

Unweighted ⇒ BFS

Acyclic/cyclic weighted
- +ve → Dijkstra
- -ve → Bellman ford (-ve wt cycle)

detect ← -ve wt cycle

Single src Algorithms

- → Dijkstra → $O(E+V)$ → $O(V^2+V)$ ⇒ $O(V^2)$
- → Bellman ford → $O(E \times V)$ → $O\left(\frac{V(V-1)}{2} \cdot V\right)$ ⇒ $O(V^3)$

Multiple src algo

- → Floyd Warshal ⇒ $O(V^3)$

$V = \{v_1, v_2, v_3\}$

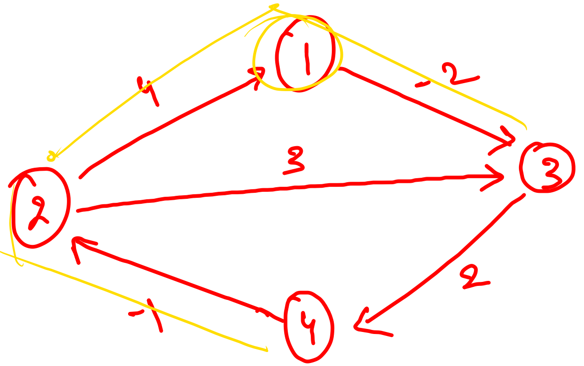
	v_1	v_2	v_3
v_1	0	–	–
v_2	–	0	–
v_3			0

Floyd → Adyacency Matrix

Multi src

$O(V \cdot V^2)$

$O(V^3)$



$K = 2$

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & \infty & -2 & \infty \\
2 & 4 & 0 & 2 & \infty \\
3 & \infty & \infty & 0 & 2 \\
4 & \cancel{\infty}\ 3 & -1 & \cancel{\infty}\ 1 & 0
\end{array}
$$

$K \Rightarrow 1, 2, 3, 4$

```
for(k ----> vtces){
    for(i ----> ){
        for(j -> ){

        }
    }
}
```

$dp[i][j] = Math.min(dp[i][j], dp[i][k] + dp[k][j])$

$spw[4 \to 3]$

$(pw[4 \to 3]$ $\quad pw[4 \to 2] + pw[2 \to 3]$

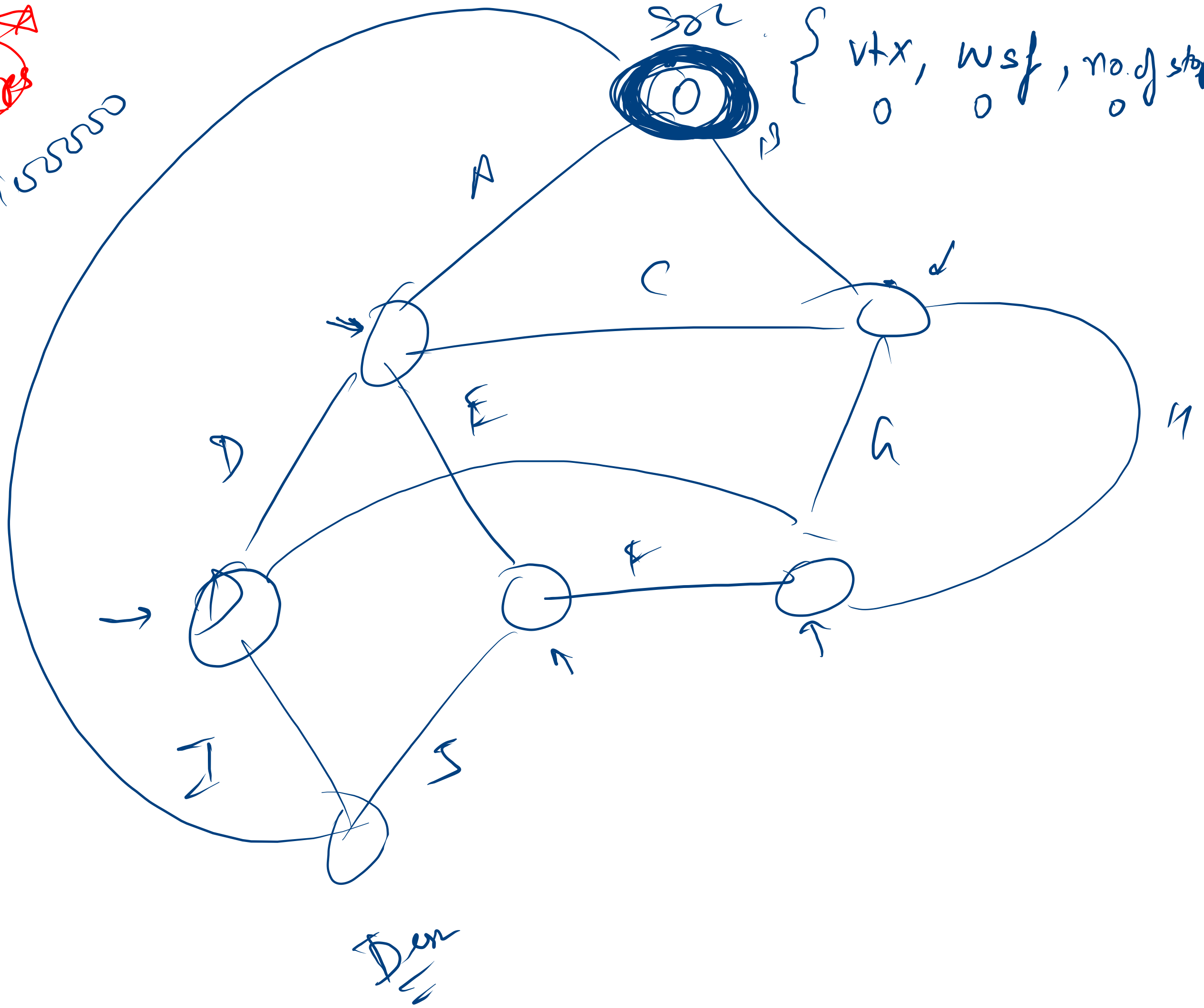
∞ $\quad -1 + 2$

1

Dijkstra

↳ no. of stops

1000000

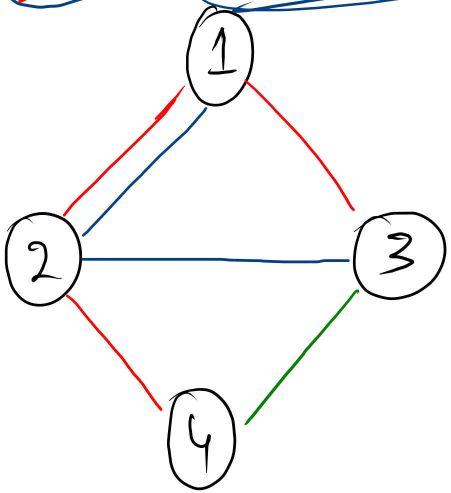


{ vtx, wsf, no.of stops }

0, 0, 0

```
while(PQ.size > 0){
    remove
    if (no.of stops > K)
        ↳ continue;
    if(src == dest){
        return wsf;
    }
}
```

DSU + Sort ⇒ O(E log E)



1	2	3
2	3	3
~~1~~	~~2~~	~~1~~
~~1~~	~~3~~	~~1~~
2	4	1
3	4	2

T_1 → Alice

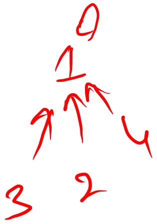
T_2 → Bob

T_3 → anyone

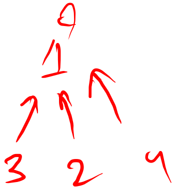
$\boxed{\text{Count} \Rightarrow 0+1+1+1+1 \Rightarrow 4}$

use

Alice T_1



bob T_2



Ans ⇒ T.E. − U.E.

⇒ 6 − 4

$\boxed{\text{Ans} \Rightarrow 2}$

2