2019038004 조민우

"저는 이 시험에서 부정행위가 발각시 어떠한 조치를 취하더라도 해당조치에 모두 따르겠습니다." (서명)

1. $f(n) = 100n + \log n$. $\quad g(n) = n + (\log n)^2$.

n vs $\log n$. $\quad$ n vs $(\log n)^2$.

n에 2^t 대입 $\quad$ n에 2^t 대입

$\Rightarrow 2^t$ vs t. $\quad$ $\Rightarrow 2^t$ vs t^2.

2^t dominate t. $\quad$ 2^t dominate t^2

$\therefore f(n) = O(n)$ $\quad$ $\therefore g(n) = O(n)$

$\therefore f(n) = O(g(n))$, $g(n) = O(f(n))$ $\Rightarrow f(n) = \Theta(g(n))$. (참).

2. Array $A[1 \text{ to } n]$, $i>j$. $A[i]<A[j]$: inversion.
Merge Sort를 진행하며 Merge 하는 과정중에서, Sorted된 divided component 2개를 Merge 해야할때 각각을 x, y로 본다. 이때, x와 y를 merge 할때 index가 변경된 개수를 count 하고 1을 뺀다. 이를 반복한다.

ex) 3 7 2 8 6 5 9.
3 | 2 7 | 6 8 | 5 9
①, ①
2 3 7 | 5 6 8 9
①, ②
2 3 5 6 7 8 9
②
⑦

Time Complexity : Merge Sort($O(n\log n)$) 중 merge 도중 count의 변수값을 증가시키는 연산($O(1)$)을 실행. 이때 이 연산의 worst case 실행횟수는 $n\log n$번

$\therefore$ Time Complexity $= O(n\log n)$

3. Selection using median of medians. sub size = 7.

Median $O(1)$ subarray $\frac{n}{7}$

subproblem 최대개수 : $n - (\frac{4}{7} \times \frac{1}{2})n = \frac{5}{7}n$.

$\therefore T(n) = T(\frac{n}{7}) + T(\frac{5}{7}n) + O(n)$

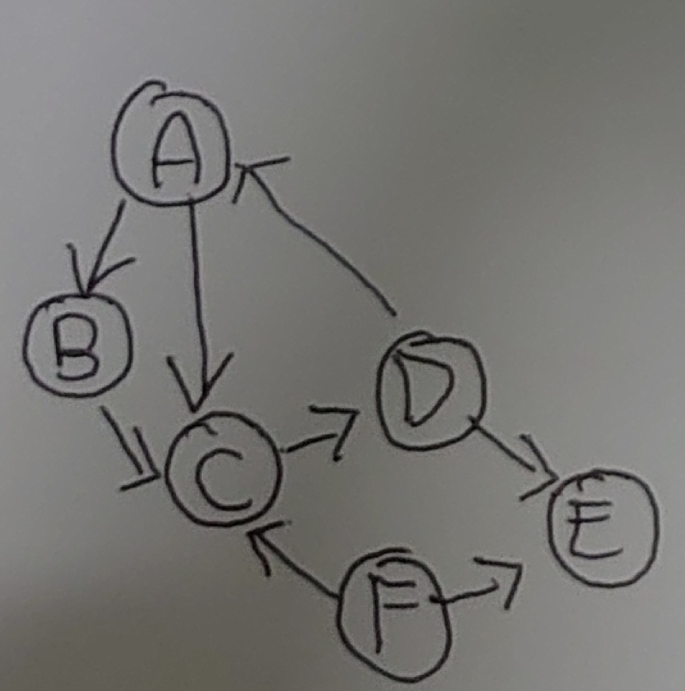
$\Rightarrow \fallingdotseq T(\frac{6}{7}n)$.

Master Theorem : $a=1$ $b=\frac{6}{7}$ $d=1$.

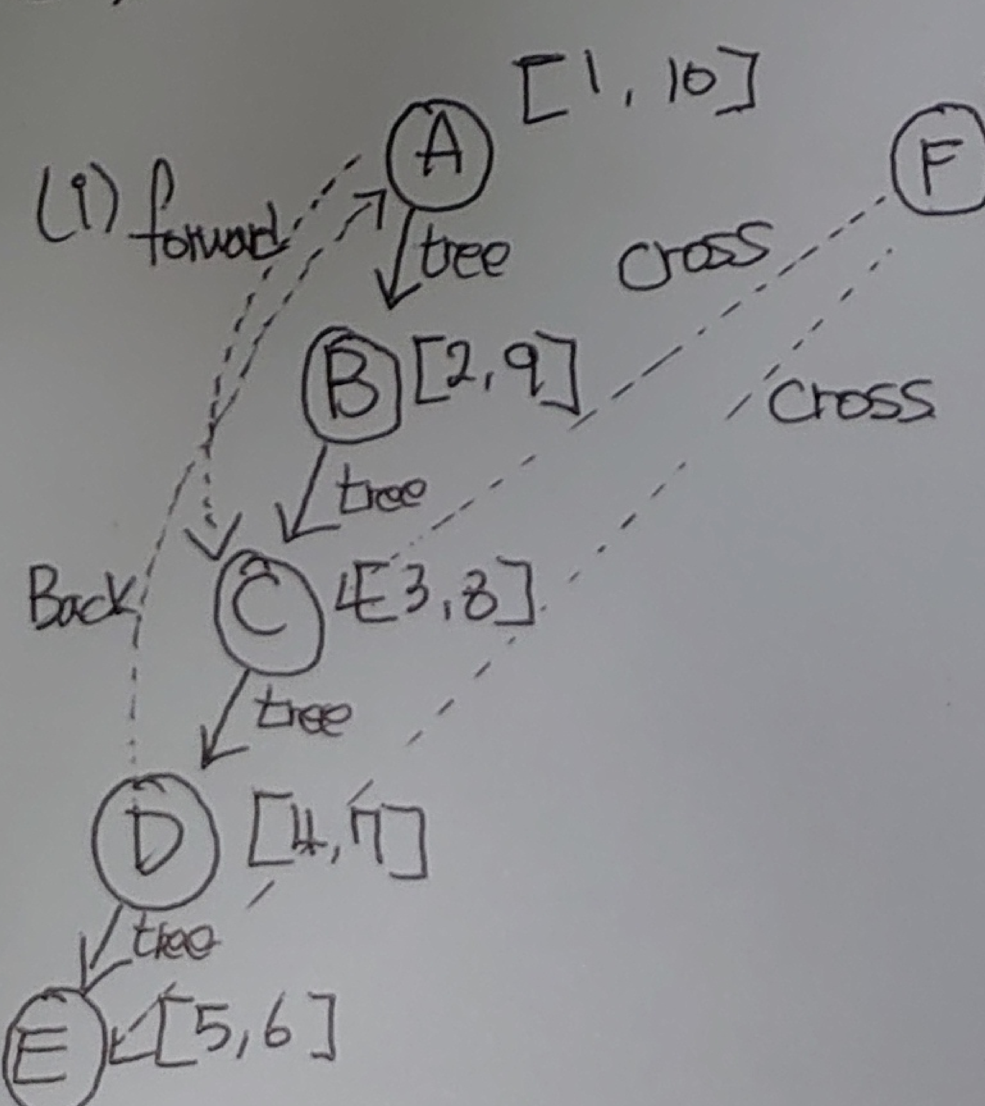
$\log_{\frac{6}{7}} 1 < 1$

$\therefore O(n)$.

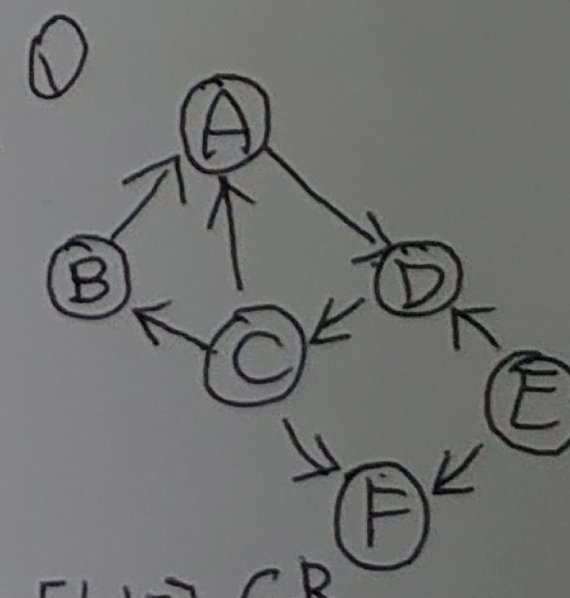
4.



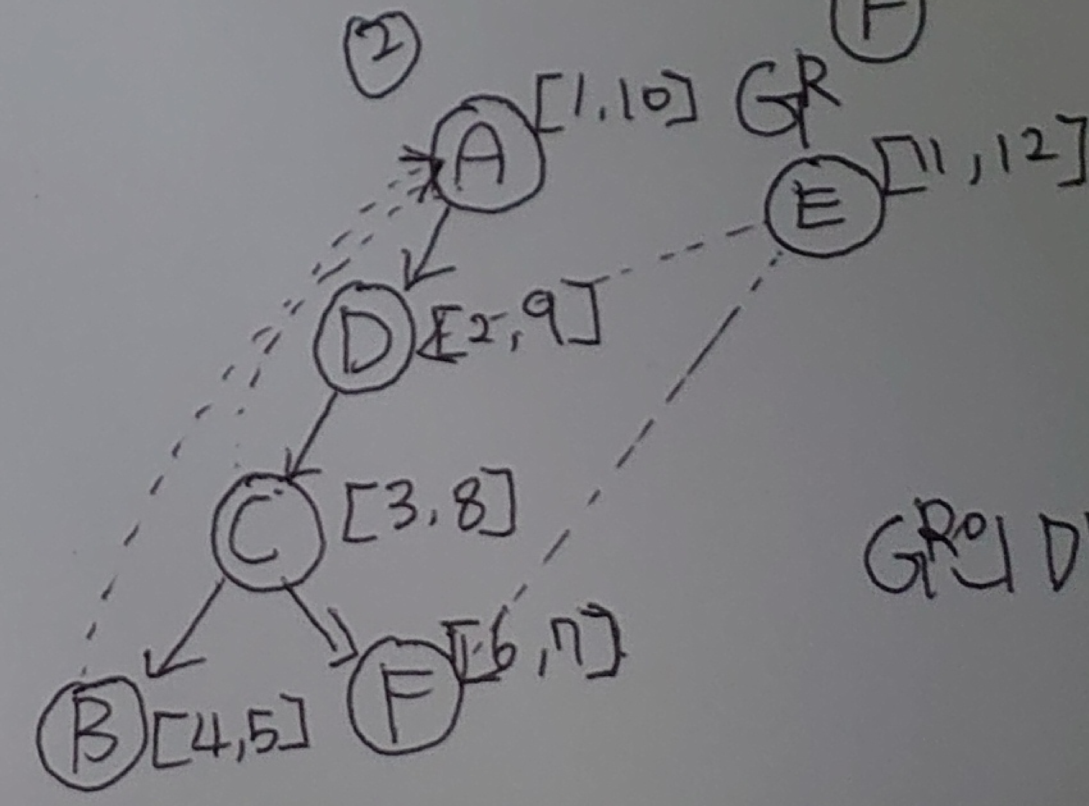
G



(ii) ①

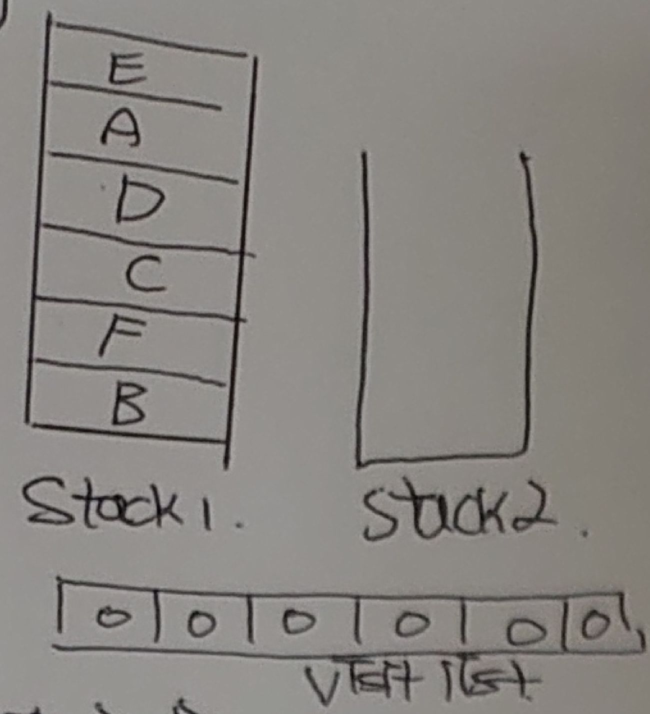


②



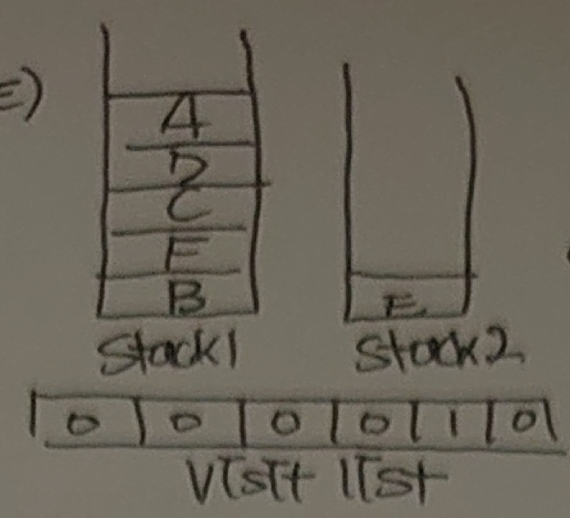
GR의 DFS Tree.

③



④

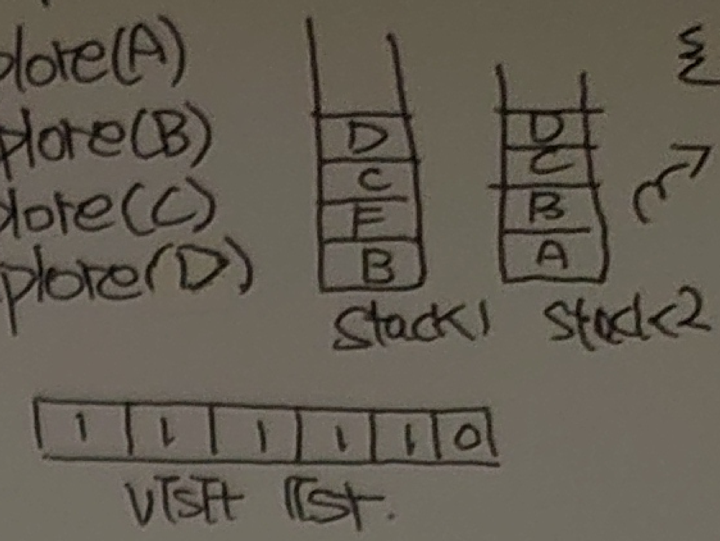
pop → E
explore(E)



→ {(E)}

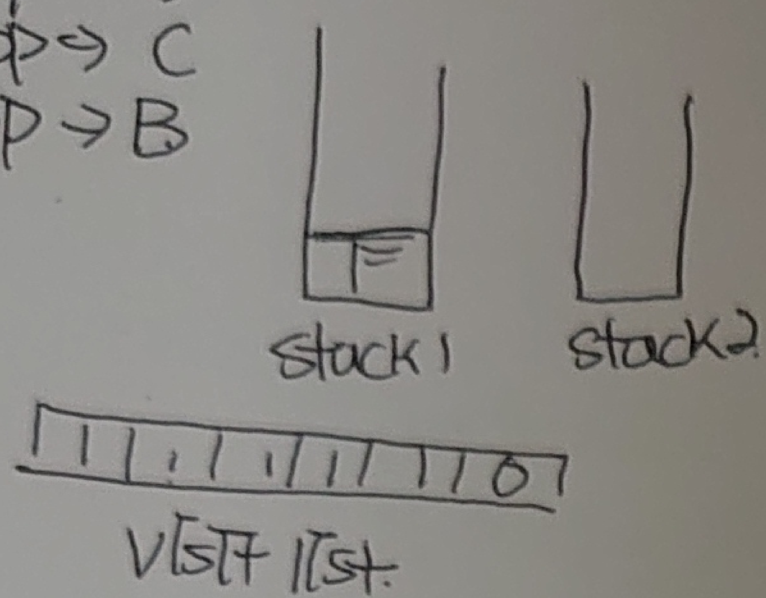
⑤

pop → A
explore(A)
explore(B)
explore(C)
explore(D)



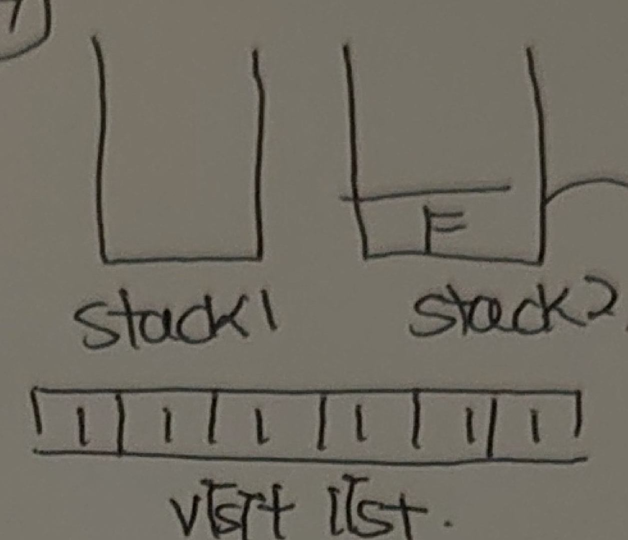
→ {(E), (A, B, C, D)}

⑥ pop → D
pop → C
pop → B



Pop → F
explore(F)

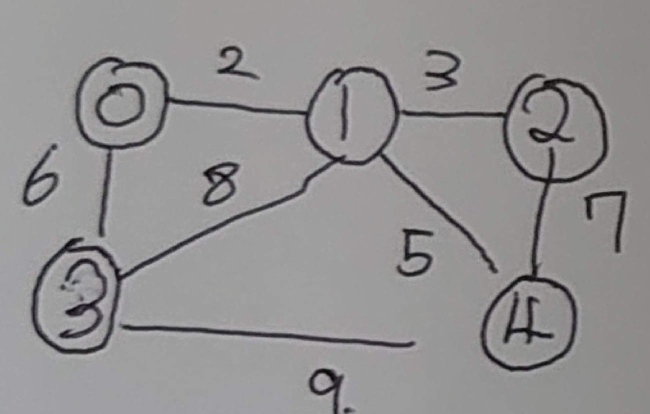
⑦



→ {(E), (A, B, C, D), (F)}

∴ Strongly Connected Components = {(A, B, C, D) (E), (F)}

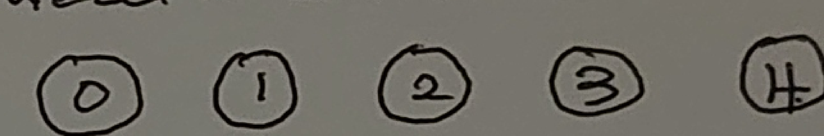
5.



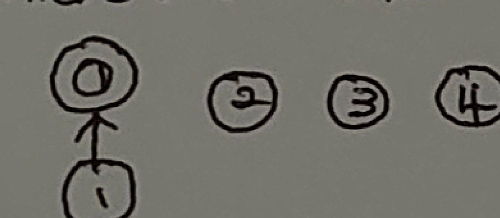
Edge-cost list (sorted)

(0, 1)	2
(1, 2)	3
(1, 4)	5
(0, 3)	6
(2, 4)	7
(1, 3)	8
(3, 4)	9

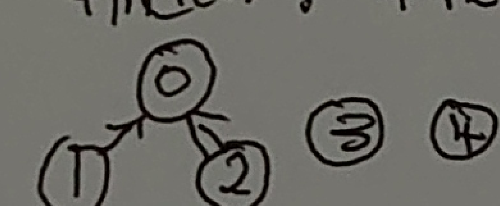
① make set for each vertex.



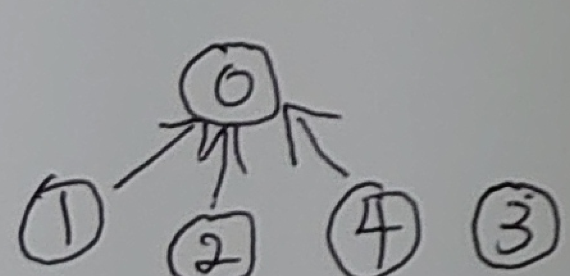
② for E(0, 1)
find(0) != find(1) union(0, 1) add(0, 1)



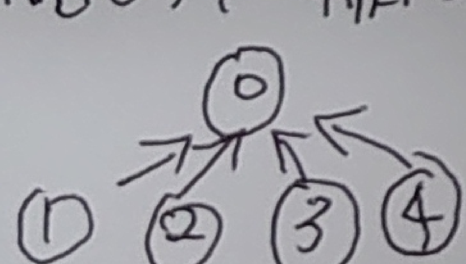
③ for E(1, 2)
find(1) != find(2) union(1, 2) add(1, 2)



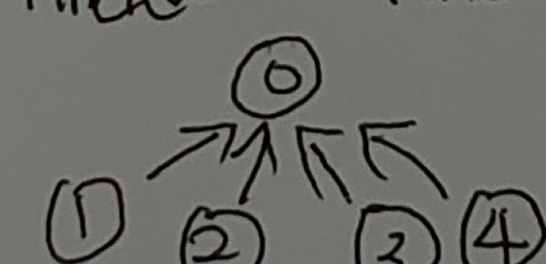
④ for E(1, 4)
find(1) != find(4) union(1, 4) add(1, 4)



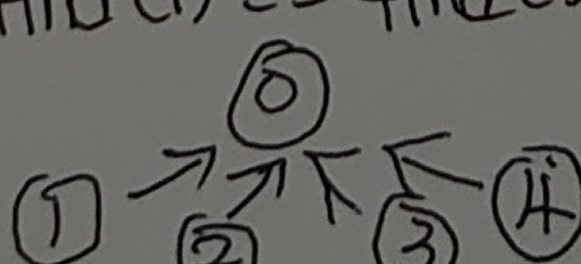
⑤ for E(0, 3)
find(0) != find(3) union(0, 3) add(0, 3)



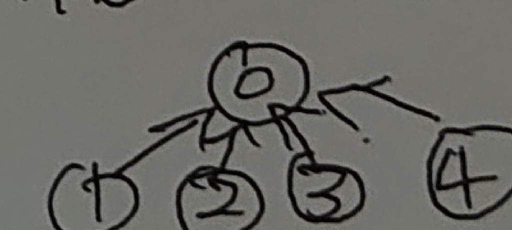
⑥ for E(2, 4)
find(2) == find(4)



⑦ for E(1, 3)
find(1) == find(3)



⑧ for E(3, 4)
find(3) == find(4)



∴ Result MST
= {(0, 1), (0, 3), (1, 2), (1, 4)}

6. cut property : MST의 edgeset T의 어느 e라도 (u, v)
그 e는, vertex u를 포함한 어떤 vertex set S와 v를 포함한 V(G)-S를 이어주는 edge들
중에서 가장 cost가 낮다.

ex) vertex set S = {0, 1, 2, 4}
V(G)-S = {3}
S와 V(G)-S를 이어주는 edge : {(0, 3), (1, 3), (4, 3)}
cost : 6 8 9
lowest!

7. Interval Scheduling $A = \{i_1 \sim i_k\}$ $O = \{j_1 \sim j_m\}$

prove $1 \le r \le k$. $f(i_r) \le f(j_r)$

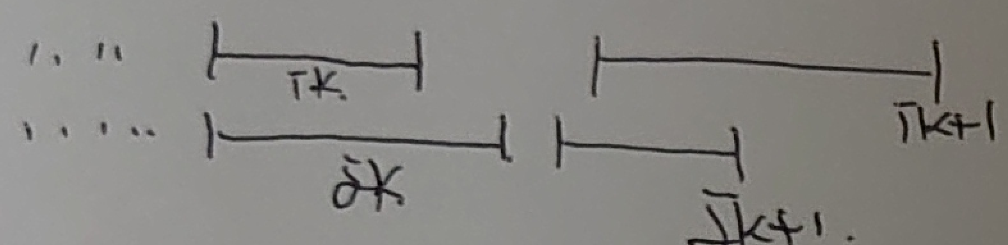
use Induction on k

i) $k = 1$. $f(i_1) \le f(j_1)$

$\because$ 종료시간이 가장 빠른 pick $\therefore f(i_1)$

ii) $r = k$ $f(i_k) \le f(j_k)$이 성립한다고 가정.

iii) $r = k+1$ 일때 $f(i_{k+1}) > f(j_{k+1})$ 이라면



j_{k+1}의 시작시간 $(s(j_{k+1}))$은 $f(i_k)$보다
클 수 밖에없고, 이때 $f(i_{k+1})$이 $f(j_{k+1})$보다 크다면,
우리는 greedy algorithm 정의상 사용했는 feasible 한 job들중
끝나는 시간이 가장 빠른것을 pick 한다는 전제에 위배된다. (i_{k+1} 대신 j_{k+1}을 pick해야한다.)

$\therefore f(i_r) \le f(j_r)$은 성립한다.