P = 0 inversions and no idle time.

- Greedy output has property P
- If two schedules have property P, then they have the same ~~min~~ max lateness. ∴ All schedules with P have the same max lateness.

Today: lemma 2: There exists a schedule with property P that is optimal.

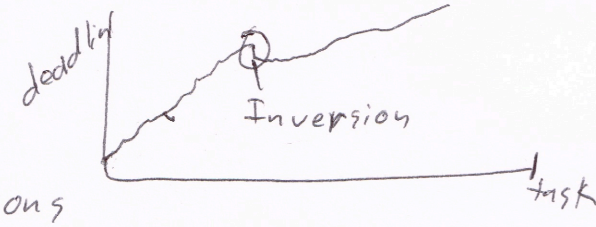
Proof of Lemma 2: By contradiction

- Give me a schedule O and claim that's it's optimal, and that it contains idle and inversions.
- For contradiction, We'll show that it can be improved by adding ~~propety~~ property P.

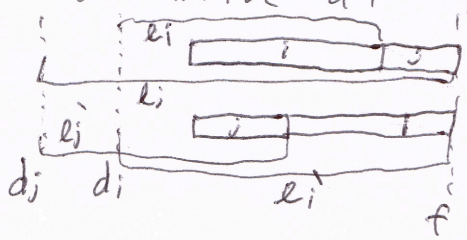
remove idle time: This can only improve the schedule. we want to minimize $\ell_i = \max(f(i) - d_i, 0)$. if f(i) is decreasing, ℓ_i must decrease or stay the same.

remove inversions: Any where I find an inversion I "fix it" to improve the schedule.

- If there is an inversion, there must exist two consecutive tasks that are inverted.
- Swap the consecutive tasks and show an improvement.
- can do this to fix all inversions and arrive at P with $\leq$ max ℓ_i



tasks i, j



$d_j < d_i$

$L = \max(f(i) - d_i, f(j) - d_j)$

(ℓ_i, ℓ_j)

$L' = \max(\ell_i', \ell_j)$

$L = \max(f - t_j - d_i, f - d_j) = f - d_j$

$f - (t_j + d_i)$

$t_j + d_i > d_j \quad \therefore \; f - d_i > f - t_j - d_i$

$L' = \max(f - d_i, f - t_i - d_j) < f - d_j$

$\therefore L' < L$ which implies fixing the inversion improved the schedule if the tasks were late and stayed the same if they were both on time.

Shortest Path Problem

Input: Directed Graph $G=(V,E)$
integers $\ell_e \geq 0$ $\forall e \in E$ (Edges have weights)
$s \in V$ (starting node)

Output: $\forall t \in V$, the length of the shortest path from s to t (s-t path)

- Given a path, $\ell(P) = \sum_{e \in P} \ell_e$

for all s-t paths P return $\min_P \ell(P)$

Special Case: $\ell_e = C$ $\forall e \in E$ for some constant $C > 0$

this can be solved by BFS in $O(m+n)$ time

Dijkstra's Algorithm

Instead of shortest path, output $d(t)$ = the length of the shortest s-t path.

main idea:

- greedy algorithm
- At each iteration we find one edge/node in the path and never change it.

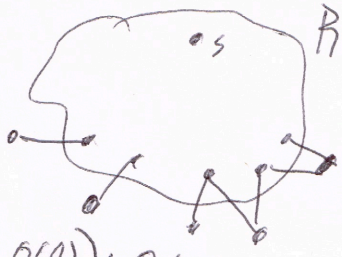
1. $d(s)=0$; $R=\{s\}$
2. while $\exists (u,x)\in E$ s.t. $u\in R \wedge x\notin R$
   - 2a. Pick x with the smallest $d'(x)$
   - 2b. $R=\{x\}\cup R$
   - 2c. $d(x)=d'(x)$;
3. return d's

$$d'(x)=\min_{u}(d(u)+l_e) \qquad e=(u,x)$$

node	s	t_1	t_2	. . .	t_{n-2}	t_{n-1}
d	$d(s)$	$d(t_1)$	$d(t_2)$		$d(t_{n-2})$	$d(t_{n-1})$

runtime:

1. $O(1)$
2. $O(n)$
   - 2a. $O(m)$
   - 2b. $O(1)$
   - 2c. $O(1)$
3. $O(1)$



$$T(n)=O(1)+O(n)\big(O(m)+O(1)+O(1)\big)+O(1)$$
$$=O(n\cdot m)$$

Let P_u be the path from s to u during the run of Dijkstra.

Proof of correctness of Dijkstra's algo:

Let R be the set at any point in the run of the algo, then P_u is a shortest path from s for any $u \in R$.

At termination, $R = V$ ∴ Dijkstra's implied to be correct, R is the connected component of s

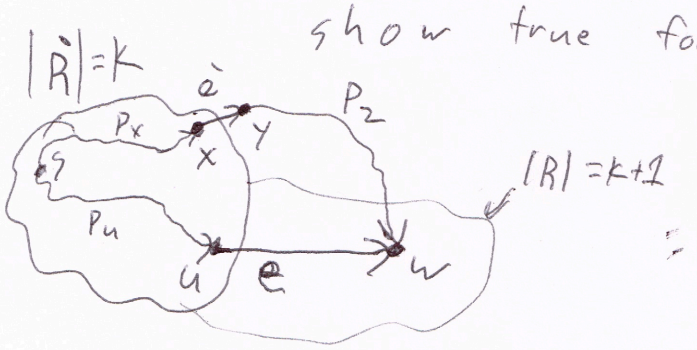
By Induction!... on $|R|$

Base case: $|R| = 1$ (only s in R)

$P_s = 0$. can't get better than that, ✓

Inductive step: assume true for $|R| = k$

show true for $|R| = k+1$



for contradiction, assume $d(x) + \ell_e + |P_2| < d(u) + \ell_e$

$|P_x| + \ell_e < |P_u| + \ell_e$

$d'(y) < d'(w)$

∴ the algo would add y to R instead of w

$O(m \log n)$ implementation

Use a heap to store the $d'(v)$ values.

fix up and extract take $O(\log n)$ time

1. initialize $d'(s) = 0$ $\quad d'(v) = \infty \ \forall v \in V$
2. pop the smallest d' (call it u) $\quad d(u) = d'(u)$, add R to u
   - 2a. update all d' by looking at edges from u.
3. return d's

---

1. $O(1)$
2. $O(\log n)$
   - 2a. $O(m)$
3. $O(1)$

$$T(n) = O(1) + O(\log n) O(m) + O(1)$$
$$= O(m \log n)$$ ✓