2 a)

Given n skiers at heights P[1..n]
& n skis at heights S[1..n].

Assign skis to skiers to min
average height difference.

ski
people

P[4, 10]
S[2, 5]

~~Greedy: 1 + 8 = 9~~

OPT: 2 + 5 = 7

2b) Consider some optimal assignment A.

If it doesn't assign lowest to lowest...

Let $cost(A) :=$ total difference in heights

Say skier 1 & ski 1 are lowest.

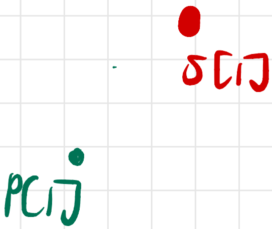
Suppose A matches skier 1 to ski j & ski 1 to skier i.

So swap those assignments.

By symmetry assume $P[1] \leq S[1]$.

If $P[i] < S[1]$, then new assignment

$\cdot S[j]$ costs



$$\begin{aligned} & cost(A) + S[j] - P[i] \\ & + (S[1] - P[1]) \\ & - (S[j] - P[1]) \\ & - (S[1] - P[i]) \\ = \; & cost(A) \end{aligned}$$

If $S[1] \le P[i] \le S[j]$, new cost is

$$
\begin{aligned}
&cost(A) + \cancel{S[j]} - P[i] \\
&\quad + S[1] - \cancel{P[1]} \\
&\quad - (\cancel{S[j]} - \cancel{P[1]}) \\
&\quad - (P[i] - S[1]) \\
&= cost(A) + 2S[1] - 2P[i] \\
&\le cost(A)
\end{aligned}
$$

If $S[j] \le P[i]$ then both pairs are better!

So we could assign lowest ski to lowest skier.

Rest of algorithm correct by induction.

S2019 - 3 b)

Given input to SSSP.

But! - no negative cycles

- shortest path from s to t uses $\leq k$ edges.

Use k iterations of main Bellman-Ford loop. We now know all shortest paths of $\leq k$ edges.

$O(kE)$ time.

S2019 - 3a) Dijkstra with

α time Insert

β time ExtractMin

γ time DecreaseKey

DIJKSTRA(s):
    INITSSSP(s)
    INSERT(s, 0)
    while the priority queue is not empty
        $u \leftarrow$ EXTRACTMIN() $\leftarrow \leq V$ times
        for all edges $u \to v$ $\leftarrow \leq E$ times
            if $u \to v$ is tense
                RELAX($u \to v$)
                if v is in priority queue
                    DECREASEKEY(v, $dist(v)$) $\leftarrow \leq E$ times
                else
                    INSERT(v, $dist(v)$) $\leftarrow \leq V$ times

Total time: $O(\beta V + \alpha V + \gamma E)$

Fibonacci heaps:

(amortized) $\alpha = O(\log n)$

$\beta = O(\log n)$

$\gamma = O(1)$

$\Rightarrow$ Dijkstra (+ Prim-Jarník): $O(V \log V + E)$

S2017-4: Given points $P \subseteq \mathbb{R}^2$

Polygonal path has vertices in P.

Is monotonically increasing is every edge goes $\nearrow$

Goal: Find longest such path.

Build a DAG $G=(V,E)$.

$V := P$

$E := \{ q \to p \mid x_q < x_p \wedge y_q < y_p \}$

$w(q \to p) := \text{Length}(q, p)$

Find longest path in G.

LPF(v): longest path length from v.

$$LPF(v) = \begin{cases} 0 & \text{if } v \text{ is a sink} \\ \max\limits_{v \to w} \left(w(v \to w) + LPF(w) \right) & \text{o.w.} \end{cases}$$

solve in postorder

return largest LPF(v)

$O(V+E)$ time

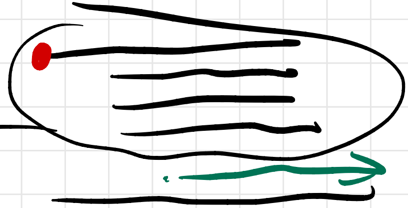
$|V| = n$

$|E| = O(n^2)$

so $O(n^2)$ time

F2019-3:

a) Wrong



b) Correct

c) Wrong (pretty sure; time is short)

d) Correct

e) Wrong