2a) Given \( n \) skiers at heights \( P[1..n] \) and \( n \) skis at heights \( S[1..n] \), assign skis to skiers to minimize average height difference.

\[
P[7, 10, 4]
\]
\[
S[2, 5]
\]

Greedy: \( 1 + 8 = 9 \)

Opt: \( 2 + 5 = 7 \)
26) Consider some optimal assignment $A$. If it doesn’t assign lowest to lowest...

Let $\text{cost}(A) = \text{total difference in heights}$.

Say skier $1 + \text{skil} 1$ are lowest.

Suppose $A$ matches skier $1$ to skil $j$, $+ \text{skil} 1$ to skier $i$.

So swap those assignments.

By symmetry assume $PC_{ij} \in SC_{ij}$.

If $PC_{ij} - SC_{ij}$, then new assignment $SC_{ij}$ costs

\[
\text{cost}(A) + SC_{ij} - PC_{ij} + (SC_{ij} - PC_{ij}) - (SC_{ij} - PC_{ij}) - (SC_{ij} - PC_{ij}) \]

\[= \text{cost}(A) \]
If \( SC_1 \leq PC_i \leq SC_2 \), new cost is

\[
\begin{align*}
cost(A) + SC_j - PC_i \\
+ SC_1 - PC_i \\
- (SC_j - SC_1) \\
- (PC_i - SC_1)
\end{align*}
\]

\[
= cost(A) + 2SC_1 - 2PC_i \\
\leq cost(A)
\]

If \( SC_j = PC_i \) then both pains are better!

So we could assign lowest skier to lowest skier.

Rest of algorithm correct by induction.
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.
Dijkstra with a time Insert
β time ExtractMin
γ time DecreaseKey

Dijkstra(s):
  InitSSSP(s)
  INSERT(s, 0)
  while the priority queue is not empty
    u ← ExtractMin()
    for all edges u→v
      if u→v is tense
        RELAX(u→v)
      if v is in priority queue
        DECREASEKEY(v, dist(v))
      else
        INSERT(v, dist(v))

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 (and Prim-Jarník): \( O(V\log V + E) \)
2017-4: Given points \( p \in \mathbb{R}^2 \), a **polygonal path** has vertices in \( p \).

Is monotonically increasing in every edge goes \( \rightarrow \)

**Goal**: Find longest such path.

**Build a DAG** \( G = (V, E) \),

\[
V := p
\]

\[
E := \{ q \rightarrow p \mid x_q < x_p \land y_q < y_p \}\]

\[
w(q \rightarrow p) := \text{Length} (q, p)
\]

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_{v \rightarrow w} (w (v \rightarrow w) + LPF (w)) & \text{otherwise} \end{cases}

solve in postorder

return largest LPF (v)

O (V + E) Time

1V1 = n

1E1 = O (n^2)

so O (n^2) Time
F2019-3:

a) Wrong

6) Correct

c) Wrong (pretty sure; time is short)

d) Correct

e) Wrong