Claim: No. of vertices in spanning tree on n-vertices with degree at least 3 is atmost $\frac{n}{2}$.

edges in spanning tree $= n-1$

$$\sum_{v \in V} \deg(v) = 2 \cdot \# \text{edges}.$$

$t_1 \leftarrow$ # vertices of deg. 1

$t_2 \leftarrow$ # vertices of deg. 2

$t_3 \leftarrow$ # vertices of deg. ≥ 3.

$$t_1 + t_2 + t_{\geq 3} = n$$

$$1 \cdot t_1 + 2 \cdot t_2 + 3 \cdot t_{\geq 3} \leq 2(n-1)$$

$$(t_1 + t_2 + t_{\geq 3}) + (t_2 + 2t_{\geq 3}) \leq 2n-2 \Rightarrow t_2 + 2t_{\geq 3} \leq n-2$$

Say $|X \cup Y| = t$. Using the claim above $\left.\begin{array}{l} |Y| < \frac{t}{2} \\ \Rightarrow k = |X| > \frac{t}{2} \end{array}\right\} |Y| < k.$

Now we just have to search for this set Y having at most k vertices.

Total running time: $\left(\sum_{i=0}^{k} \binom{n-k}{i}\right) \text{poly}(n)$. $\Big\} \to$ Brute force.

→ For every possible set $Y \subseteq V \setminus X$, of size at most k, compute the weight of MST on $\underline{X \cup Y}$. ← No. of such sets: $\sum_{i=0}^{k} \binom{n-k}{i}$

For each set, computing MST ← $O(\underbrace{\text{poly}(n)}_{E \log V})$

→ Then return the min. wt. amongst all of these

↳ Min. wt. Steiner tree w.r.t X and $G = (V, t)$.

Say $k < \frac{n}{3}$, $\left(\sum_{i=0}^{k} \binom{n-k}{i}\right) \cdot \text{poly}(n) \leq \underbrace{k \binom{n-k}{k} \cdot \text{poly}(n)}$.

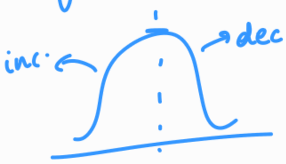
$$\left(\begin{array}{l} k < \frac{n-k}{2} \\ \frac{3k}{2} < \frac{n}{2} \Rightarrow k < \frac{n}{3} \end{array}\right)$$

$$e\left(\frac{n-k}{k}\right)^k \leq n^k$$

$\hookrightarrow n^{O(k)} \cdot \text{poly}(n)$

(Sterling's approx.)

$\because$ Binomial coeff. form bell curve



Max. value at half

Q: If e is a min. wt. edge incident on a vertex v, then argue that $e \in$ MST (Assume distint wts.).

$\because$ Cut property.

$\therefore$ min. wt. edge is part of MST.

Q. Suppose a railway station has k platforms. We are given a schedule of trains and we would like to minimise the no. of platforms used. How can we do that?.

| Interval scheduling.

But max. processes one a single processor.

Here we have multiple tracks.

$T_1, \ldots, T_m \rightarrow m$ trains

$\hookrightarrow$ One round of interval scheduling

$\downarrow$

$T_1', \ldots, T_k' \rightarrow$ Left over trains.

How to prove optimality?

Each platform has an optimal schedule.

Platforms: P_1 ——— Schedule

P_2 ———

$\vdots$

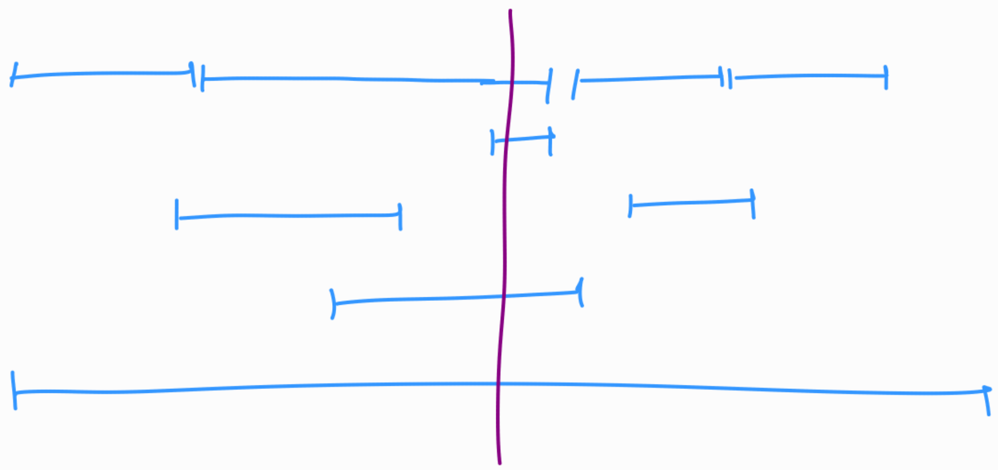
P_K. ———

Is Platform P_K needed at all? i.e., does $\exists$ a diff. scheduling of trains in P_K s.t those trains are accomodated in 1 to P_{K-1}.

No. Incompatibility is a proof

Optimality of interval scheduling problems gives optimality of whole problem.

Min. No. of platforms = Max. no. of overlaps at a given time.



Req. 4 platforms.

Complexity of algo. will only depend on comp. of sorting.

If $|\text{trains}| = n$. Then $O(n \log n)$

Q. Solved exercise (KT). of Greedy algo.

Closest pair of points.

$P_1, P_2, \ldots, P_n \in \mathbb{R}^2$. $P_i = (x_i, y_i)$

Want to find the closest pair of points.

n: power of 2.

If n is not a power of 2, nearest power of 2 is n'.

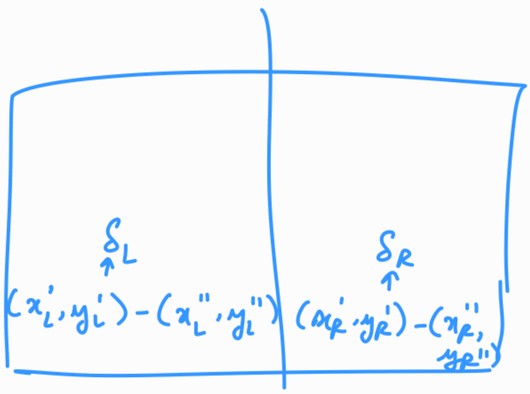
So $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \sim$

Merging.

Can't add dummy points.

$T(n') \le T\left(\frac{n'}{2}\right) + T\left(\frac{n'}{2}\right) + \sim$

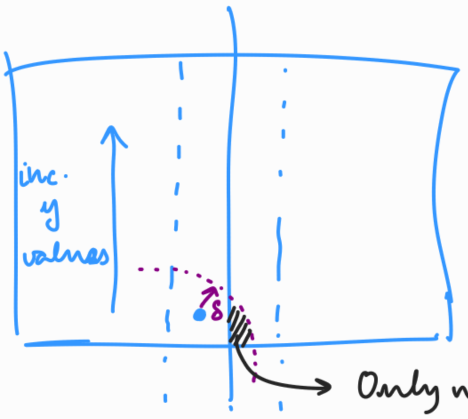
- Sort the points P_i based on x-values.
- Put a separator at the median x-value. (So divides points to roughly 2 halfs).
- Compute closest pair on each side recursively.



$\delta = \min\{\delta_L, \delta_R\}$.

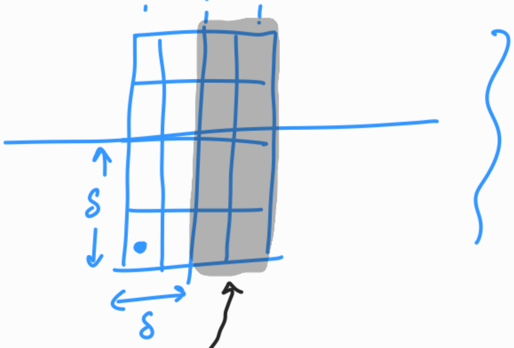
- Consider the δ-band on either side of the separator

• Consider only those points that lie in this band. (Could be the case that all pts. lie in this band).



Only need to consider these pts.

New task: Find pairs of points s.t they are not on same side of separator & have dist. $< \delta$.



In this grid, there are at most 16 points and at most one point per $\frac{\delta}{2} \times \frac{\delta}{2}$ sq.

If the point was at the bottom, then we only need to compare it with 8 other points which are located on the other side.

↳ Have to check this

(OR)

15 other points which are there in the grid.

So that we don't have to check if the points are on the same side or different side.