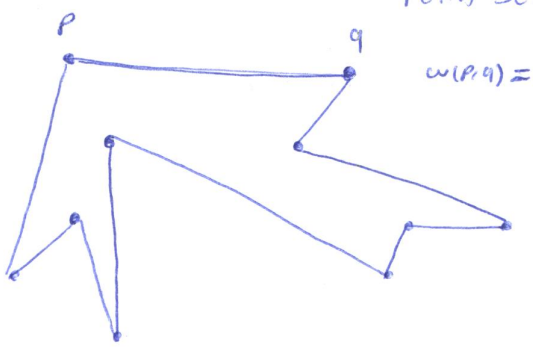
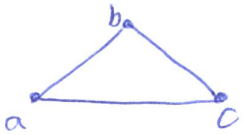
# Euclidean TSP




Point set $P$

$\omega(p,q) = |pq|$

Property 1 (Triangular Inequality):



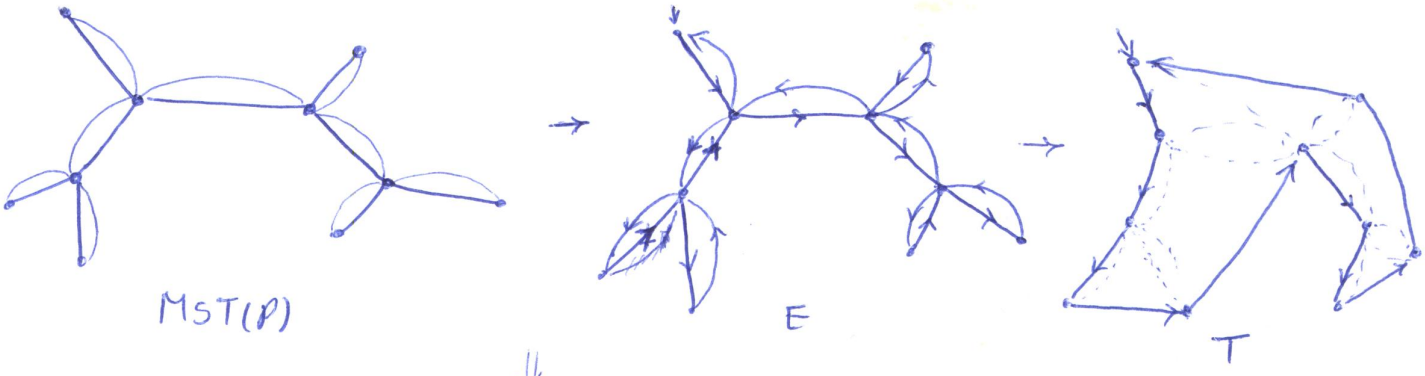

$|ac| \leq |ab| + |bc|$

Property 2 (Extention of Property 1):


$|ab| \leq |e_1| + |e_2| + \cdots + |e_m|$

## First Alg.

- Find MST($P$)
- Duplicate edges of MST($P$)
- Find an Eulerian tour $E$
- Convert $E$ to a tour $T$ by ~~go~~ walking through vertices of $E$ while skipping the vertices that already visited.




⇓

Eulerian tour: a closed walk which visits each vertex of the graph exactly once.

Theorem: a (connected) graph has an Eulerian tour iff all vertices are of even degree.

**Theorem**: The above alg. is a 2-approximation alg. for ETSP.

**Proof:**

let OPT be an optimal tour.

by removing any edge from opT we optain a spanning path T' which is a spanning tree as well. Thus

$$\left.\begin{array}{l} T' < OPT \\ MST \leq T' \end{array}\right\} \Rightarrow MST \leq OPT \quad (1)$$ abuse the notation for weight

The weight of E is two times the weight of MST because E contains two copies of each edge in MST.

$$E = 2\,MST$$

Since we obtain T by shortcutting the edges of E, the cost of T does not exceed the cost of E (remember the triangle inequality)

$$\Rightarrow T \leq E = 2MST \quad (2)$$

Finally $\xrightarrow{(1)\&(2)} T \leq E = 2\,MST \leq 2 \cdot OPT$

$$\Rightarrow T \leq 2 \cdot OPT$$

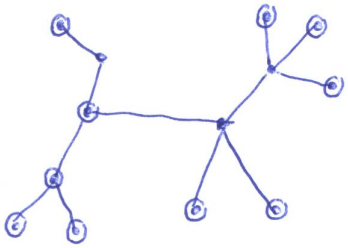
## Second Alg:

- Find MST(P)
- Let $V'$ be the vertices of odd degree in MST(P)
- Compute a minimum perfect matching M for $V'$
- Add edges of M to MST(P)
- Find an Eulerian tour E in M ∪ MST(P)
- Convert E to a tour T

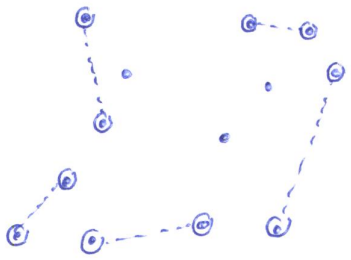
**perfect matching:** a matching which covers all the vertices

**Observation:** the number of vertices of odd degree is even.

⇒ $V'$ has a perfect matching

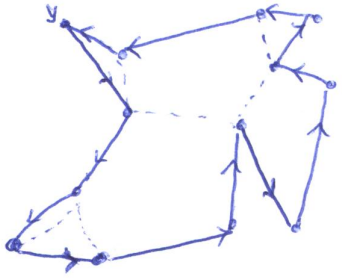


MST and $V'$



M

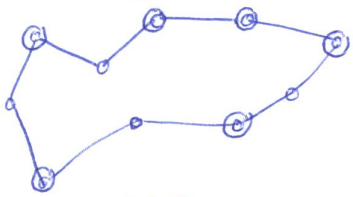
M ∪ MST is Eulerian



T

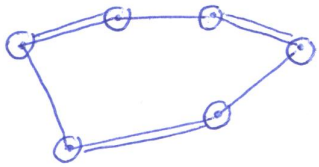
**Theorem:** The above alg. is a $\frac{3}{2}$-approximation for the ETSP.

Let OPT be ~~the~~ an optimal tour and let $T'$ be ~~the~~ tour obtaind from OPT by shortcutting the vertices of $V \setminus V'$



OPT

⇒



$T' = M_1 + M_2$

$T' \leq OPT$

$T'$ contains two perfect matching for $V'$ ~~say~~ $M_1$ and $M_2$

M is a minimum perfect matching and smaller than both $M_1$ and $M_2$

$M \leq M_1$ and $M \leq M_2 \Rightarrow 2M \leq M_1 + M_2 = T'$

$\Rightarrow$ ~~$M \leq$~~ $\Rightarrow M \leq \frac{T'}{2} \leq \frac{OPT}{2}$ ∎

$T^* \leq E = M + MST$

$E = M + MST \leq \frac{OPT}{2} + OPT = \frac{3}{2} OPT$ (3)

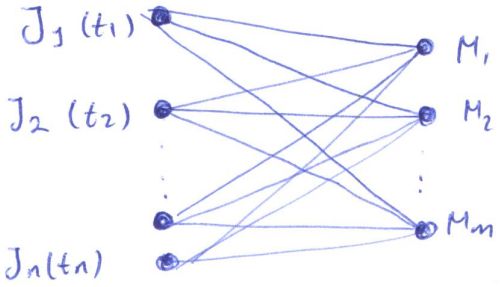
(1),(3) $\Rightarrow T \leq \frac{3}{2} OPT$

## Load Balancing:

Given $n$ jobs $j_1, \ldots, j_n$ and $m$ machines $M_1, \ldots, M_m$.

Each job $j_i$ needs time $t_i$ to be done.

Assign jobs to machines such that the time until all jobs are finished is minimized.




First Alg. (Greedy)

for $i \leftarrow 1$ to $n$
    assign $J_i$ to $M_k$ of minimum load

**Theorem:** The above alg. is a 2-approximation alg. for load balancing prob.

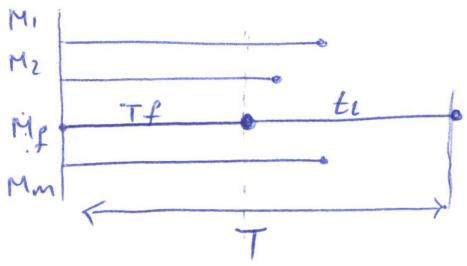
Let $T$ be the time required by alg. and $OPT$ be the optimal time.

$$OPT \geq t_{max} \quad \text{where} \quad t_{max} = \max(t_i)$$

$$OPT \geq \frac{1}{m} \sum_{i=1}^{n} t_i$$

At the end of alg., let $M_f$ be the machine which maximizes the load.

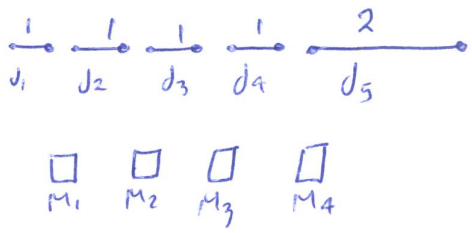
Let $J_l$ be the last job assigned to $M_f$.



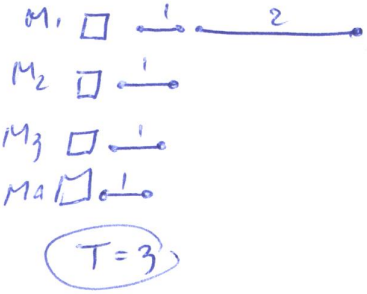

$$T = T_f + t_l \leq \frac{1}{m} \sum_{i=1}^{n} t_i + t_{max} \leq OPT + OPT$$

$$\Rightarrow T \leq 2 \cdot OPT$$

How to improve:

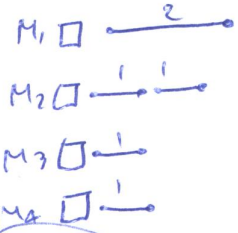



assignment 1




T = 3

assignment 2




T = 2

Observation: Its better to assign larger jobs first.

Second Alg 1.

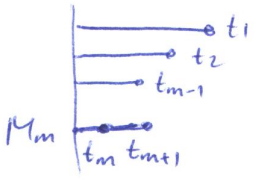
```
L ← decreasig ordered list of jobs
for i ← 1 to n
    assign L(i) to M_k of minimum load
```

Theorem: The above alg. is a $\frac{3}{2}$-approximation alg. for load balancing problem.

assume $t_1 \geq t_2 \geq \cdots \geq t_n$

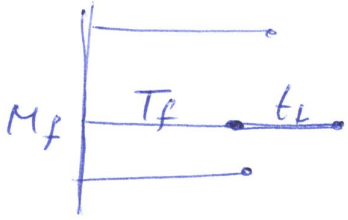
if $n \leq m$ then the alg. is optimal

if $n > m$ then $opt \geq t_m + t_{m+1}$




Let $M_f$ be the machine with maximum load and assume $J_e$ is the last job assigned to $M_f$.


$$\left.\begin{array}{l} t_i \leq t_m \\ t_i \leq t_{m+1} \end{array}\right\} \Rightarrow 2t_i \leq t_m + t_{m+1} \Rightarrow t_i \leq \frac{t_m + t_{m+1}}{2}$$



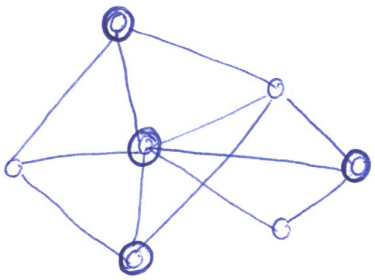

$$T = T_f + t_i \leq \frac{1}{m}\sum_{i=1}^{m} t_i + \frac{t_m + t_{m+1}}{2} \leq opt + \frac{OPT}{2}$$

$$\Rightarrow T \leq \frac{3}{2} \cdot OPT$$

## Vertex-Cover Problem

$G=(V,E)$, a vertex-cover is a subset $C \subseteq V$ s.t. for each edge $(a,b) \in E$, either $a \in C$ or $b \in C$ (or both)

C covers all the edges of G!!!



Vertex-Cover Problem: compute a cover $C^*$ of minimum size.

## First Alg.

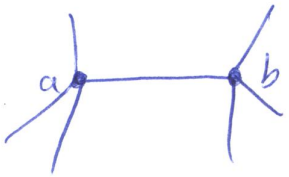
```
C ← ∅
while E ≠ ∅ do
    (a,b) ← arbitrary edge in E
    C ← C ∪ {a,b}
    E ← E \ {all edges incident on u or v}
return C
```

Running Time = $O(V+E)$

**Theorem:** The above alg. is a 2-appx alg. for Vertex-Cover problem.

Let M be the set of edges selected in the while loop.

Let $C^*$ be an optimal vertex cover.




for each edge $(a,b) \in M$, $C^*$ contains either a or b. $\Rightarrow |C^*| \geq |A|$

We add both of a and b to C $\Rightarrow |C| \leq 2|A|$

$\Rightarrow C \leq 2|A| \leq 2|C^*|$

$\rightarrow$ Any maximal matching in G is a 2-appx for Vertex-Cover Problem.

## Second Alg.

$C \leftarrow \emptyset$

while $E \neq \emptyset$ do

  $a \leftarrow$ vertex with maximum degree in current graph

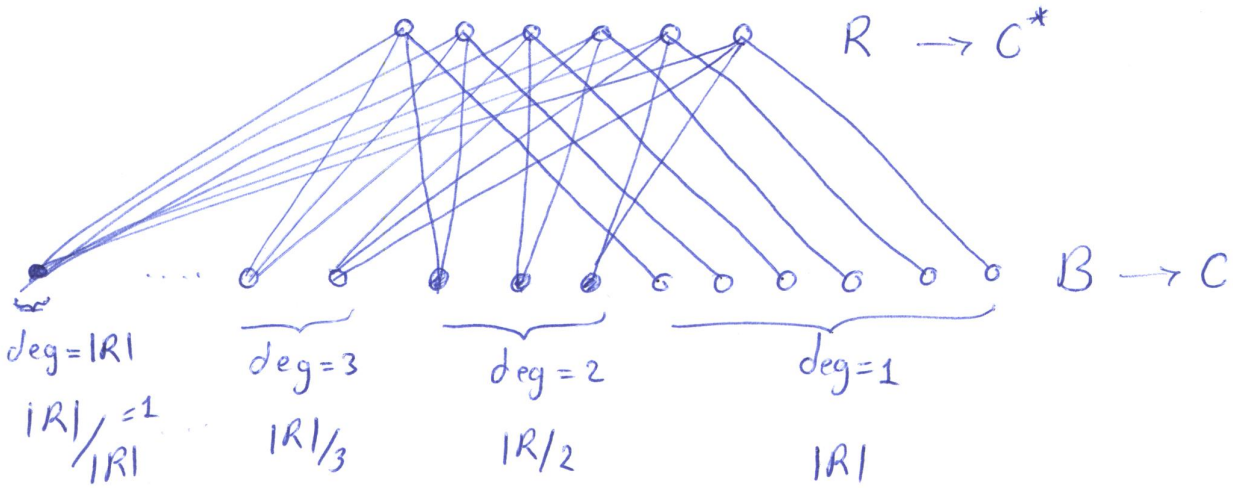
  $C \leftarrow C \cup \{a\}$

  $E \leftarrow E \setminus \{$all edges incident on $a\}$

return $C$.

What is the approximation ratio $\alpha$?

$\alpha = \Omega(\log n)$




$$|B| = \frac{|R|}{1} + \frac{|R|}{2} + \frac{|R|}{3} + \cdots + \frac{|R|}{|R|} = |R| \sum_{i=1}^{|R|} \frac{1}{i} = |R| \log |R|$$

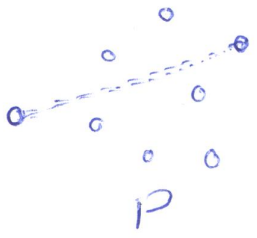
$$\left.\begin{array}{l} C = B \\ C^* = R \end{array}\right\} \Rightarrow \frac{|C|}{|R|} = \log |R| = \Theta(\log n)$$

$\Rightarrow \alpha \geq \log n$

## k-clustering:

For a point set $P$, diameter of $P$, $d(P)$, is defined as the maximum distance between any pair of points in $P$.

$$d(P) = \max\{|pq| : p, q \in P\}$$



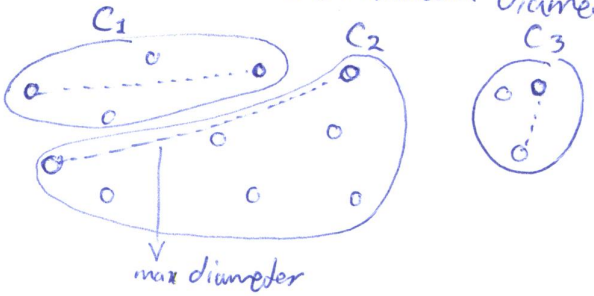
P

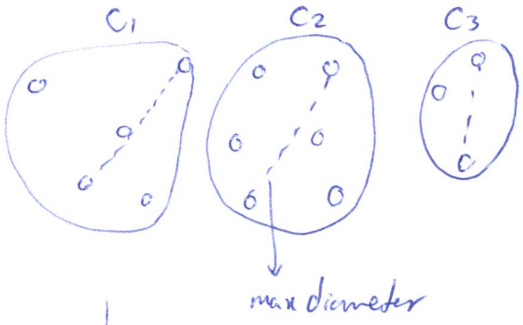
Given a point set $P$, we want to partition $P$ into $k$ clusters s.t. the diameter of clusters is minimized.

=> we want to minimize the maximum diameter.

example:

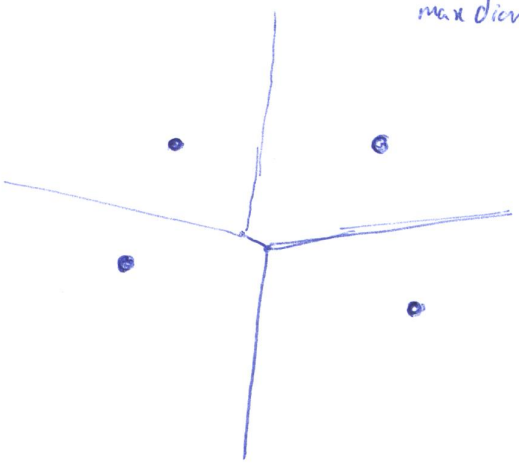
$k = 3$




## Voronoi Diagram - VD

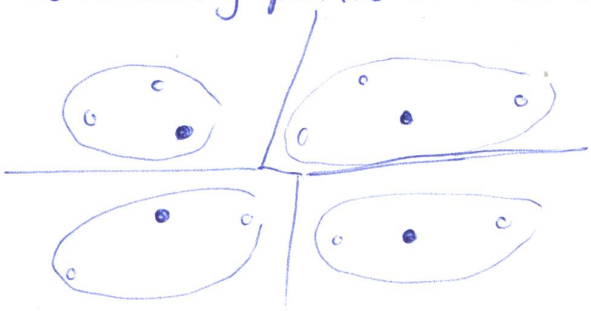
given a point set $S$, $VD(S)$ is defined as depicted in the picture.



## Algorithm

Idea: - pick $k$ points of $P$ as $S$

- compute $VD(S)$
- assign the remaining points to each voronoi cell



## Algorithm

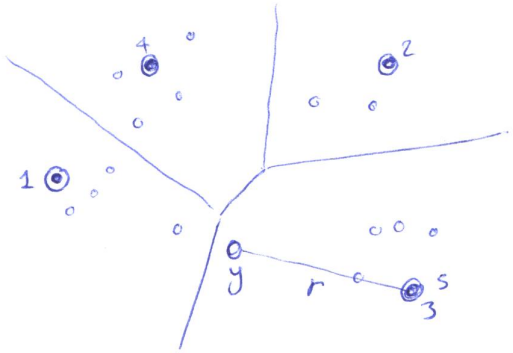
$S \leftarrow$ {arbitrary point of $P$}

for $i \leftarrow 2$ to $k$

  $x \leftarrow$ point of $P$ which is farthest from $S$

  $S \leftarrow S \cup \{x\}$

Assign the points in $P \setminus S$ to their closest point in $S$.




**Theorem:** The above alg. is a 2-appx alg. for the k-clustring problem.

Let $y$ be the next element that we have chosen by repeating the for loop ($(k+1)^{th}$ element) and let $r$ be the distance of $y$ to its closest point in $S$.

All points in cluster $s$ are at distance at most $r$ of $s$, then the diameter of $s$ is at most $2r$.

We have $k+1$ points all of them are at distance at least $r$ from each other (in each iteration the farthest point distance is decreasing) and by pegion hole principle two of them fall in the same cluster in OPT. So OPT has diameter at least $r$.