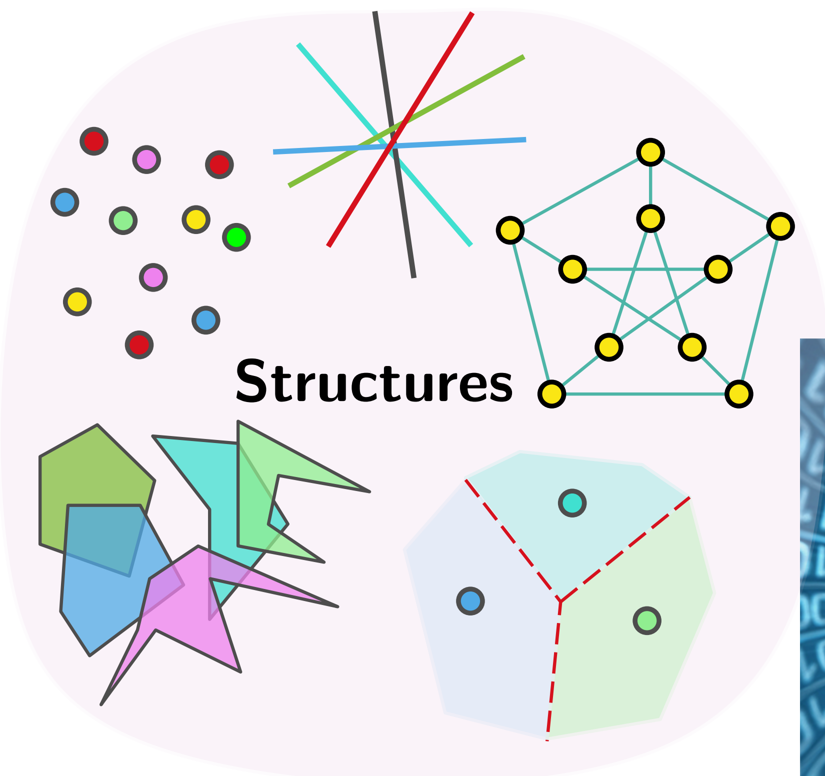
Computing: Structures; Combinations; Algorithms

Sujoy Bhore

Indian Institute of Technology Bombay

Computing ...



$$f_{k-1} = \sum_{i=k}^{d} (-1)^{d-i} \binom{i}{k} f_{i-1}\,.$$

$$N(\mathcal{A}) = \textstyle\sum_{x \in L(\mathcal{A})} (-1)^{r(x)} \mu(0,x)$$

$$\#(tP \cap \mathbb{Z}^n) = \sum_{i=0}^{d} a_i t^i,$$

$$\chi_T(\phi+2) = (\phi+2)\,\phi^{3V(T)-10}\,(\chi_T(\phi+1))^2$$

$$\chi(KG_{n,k}) = n - 2k + 2$$

$$\frac{2(2n)!3^n}{n!(n+2)!}$$

$$\frac{\lambda_1 \lambda_2 \ldots \lambda_{n-1}}{n}$$

$$\frac{\alpha(G)}{n} \leq \frac{\lambda_{\min}}{d - \lambda_{\min}}$$

$$(1-\lambda_2)/2 \leq h(G) \leq \sqrt{2(1-\lambda_2)},$$

$$\Theta(C_5) = \sqrt{5}.$$

$$\mu_c := \lim_{n\to\infty} c_n^{1/n} = \sqrt{2+\sqrt{2}}.$$

$$\ell_n \sim 2\sqrt{n}.$$

$$\lim_{n\to\infty} \mathrm{Prob}\left(\frac{\mathrm{is}(\sigma) - 2\sqrt{n}}{n^{1/6}} \leq t\right) = F(t)$$

$$b \geq v.$$

$$\mathrm{per}(A) \geq \frac{n!}{n^n}$$

$$W(C^{\perp}; x, y) = \frac{1}{|C|} W(C; y-x, y+x).$$

$$|C| \leq \sum_{i=0}^{d} \binom{n}{i},$$

$$(2/e)(1+o(1))k2^{k/2} \leq r(k+1,k+1) \leq k^{-C\log k/\log\log k}\binom{2k}{k}.\ ^{*}$$

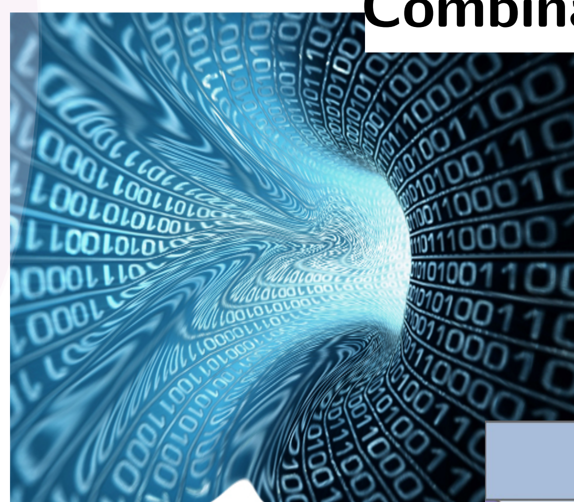
$$\aleph_\omega^{\aleph_0} < \max\{\aleph_{\omega_4}, (2^{\aleph_0})^+\}, \ ^{*}$$

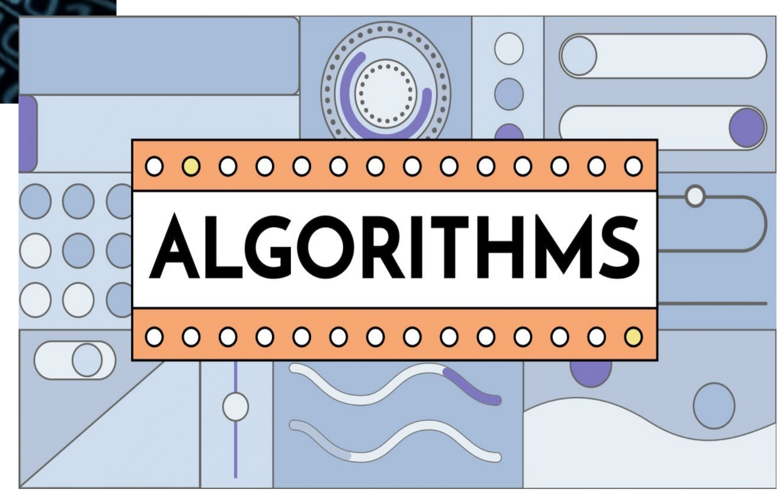
$$\max(|A+A|, |A \times A|) \geq \frac{1}{2}|A|^{4/3}(\log|A|)^{-1/3}\ ^{*}$$

$$\aleph_0 \leq^{*} \mathfrak{p} \iff \aleph_0 \leq 2^{\mathfrak{p}}$$

Combinations

$$Q_n = 2n(\ln n + \gamma - 2) + 2\ln n + 2\gamma + 1 + O\left(\frac{1}{n}\right)$$



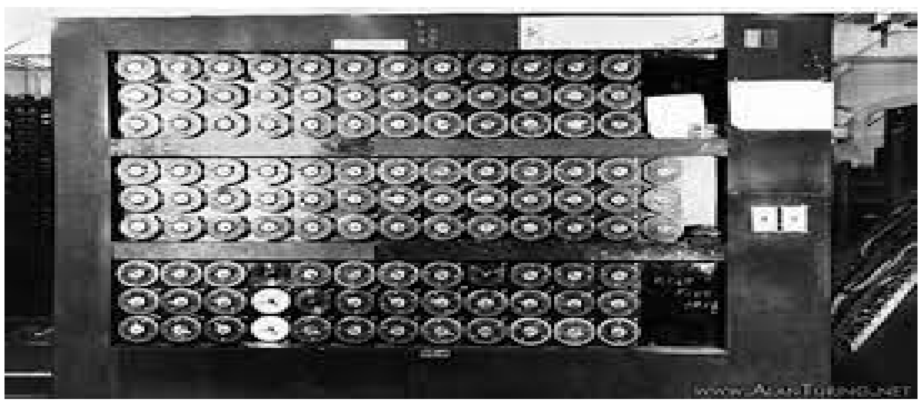


Computing ...

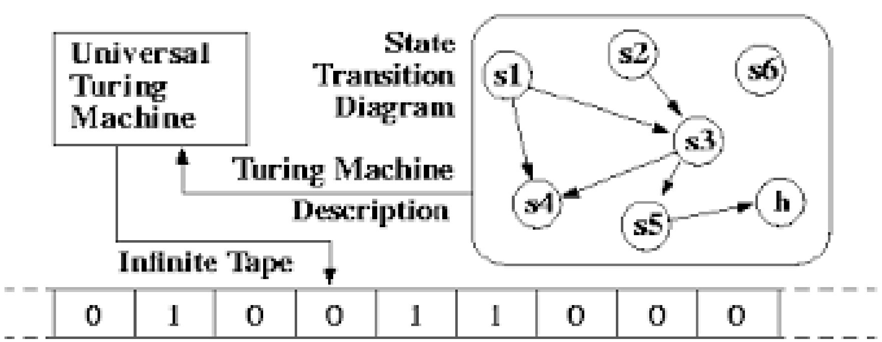
Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do... – Dijkstra/ folklore ... ?

Computing ...

Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do... – Dijkstra/ folklore ... ?

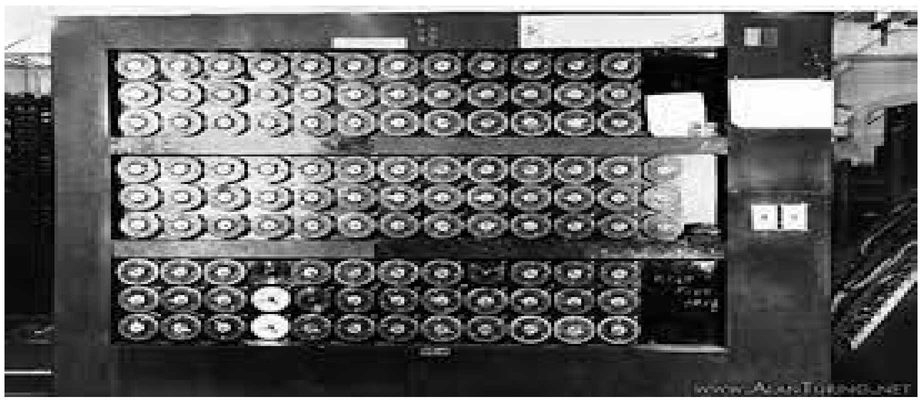


Turing era ...

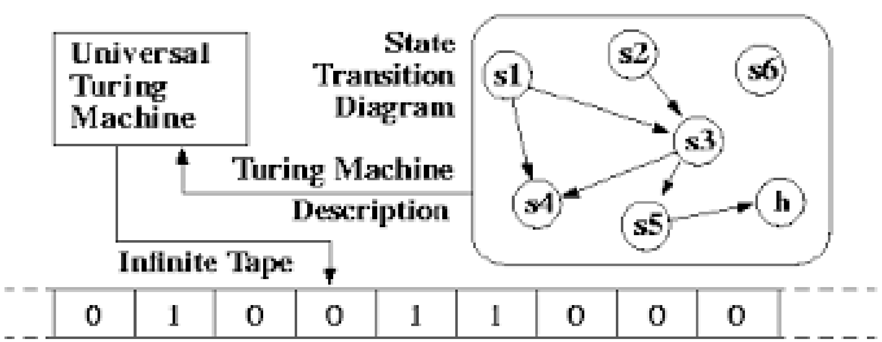


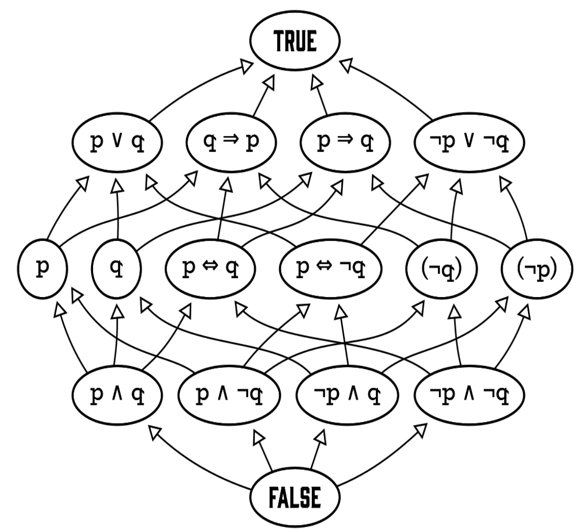
Computing ...

Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do... – Dijkstra/ folklore ... ?



Turing era ...





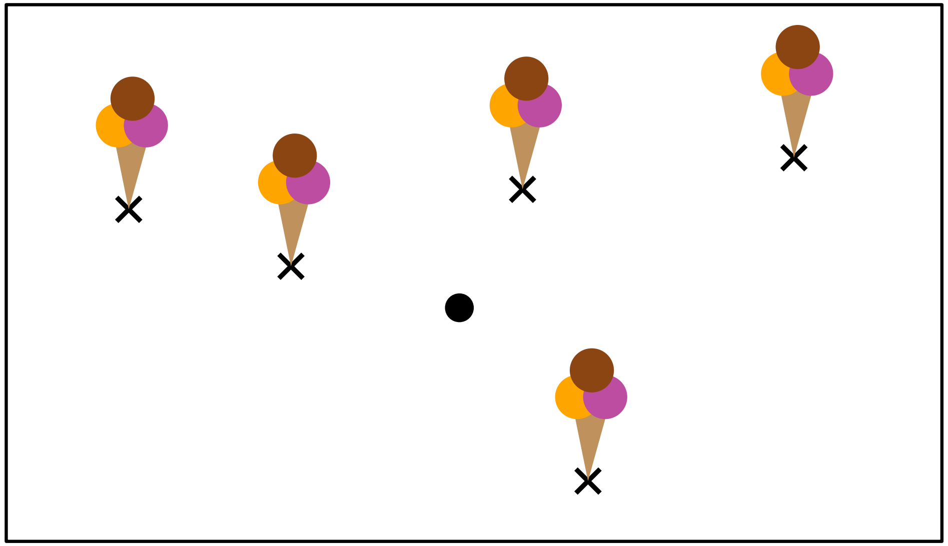
we are building upon the 2500 yrs. of knowledge ...

Problem 1

It's one of the last sunny days before winter.

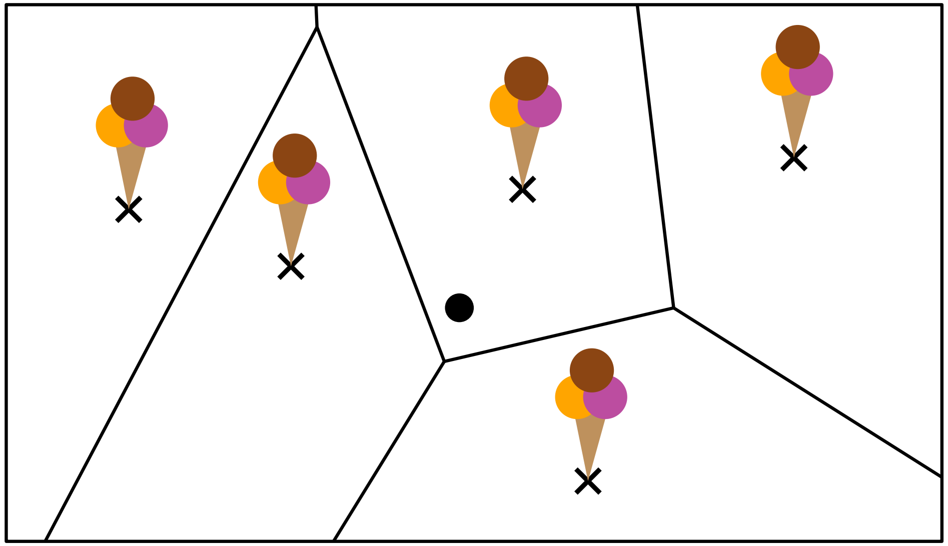
Suppose you know the location of five ice cream shops in the city.

How can you determine the closest one for any location on a map?



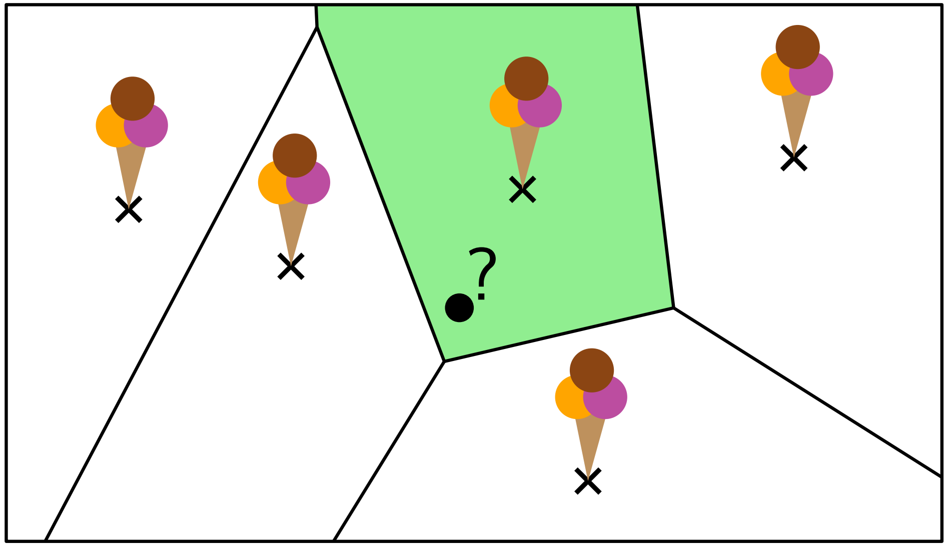
Problem 1

It's one of the last sunny days before winter.
Suppose you know the location of five ice cream shops in the city.
How can you determine the closest one for any location on a map?



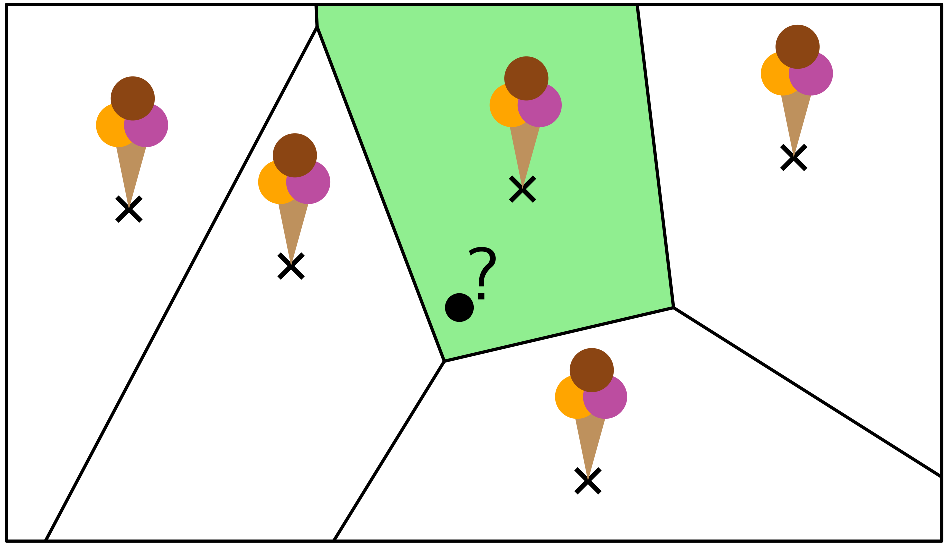
Problem 1

It's one of the last sunny days before winter.
Suppose you know the location of five ice cream shops in the city.
How can you determine the closest one for any location on a map?



Problem 1

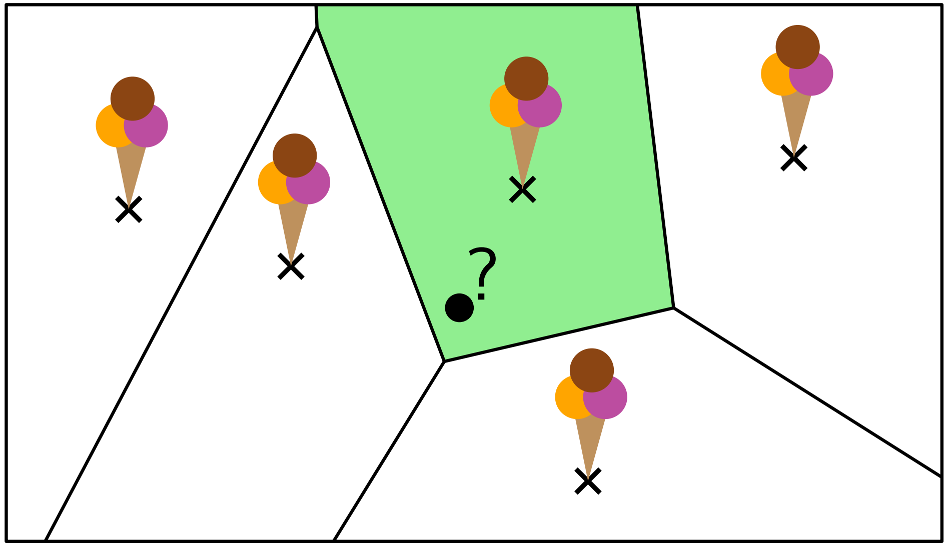
It's one of the last sunny days before winter.
Suppose you know the location of five ice cream shops in the city.
How can you determine the closest one for any location on a map?



The solution is a *subdivision* of $\mathbb{R}^2$, called **Voronoi Diagram**.....
Lots of applications, e.g.: site planning, facility placement, nearest-neighbor finding, wireless networks ...

Problem 1

It's one of the last sunny days before winter.
Suppose you know the location of five ice cream shops in the city.
How can you determine the closest one for any location on a map?

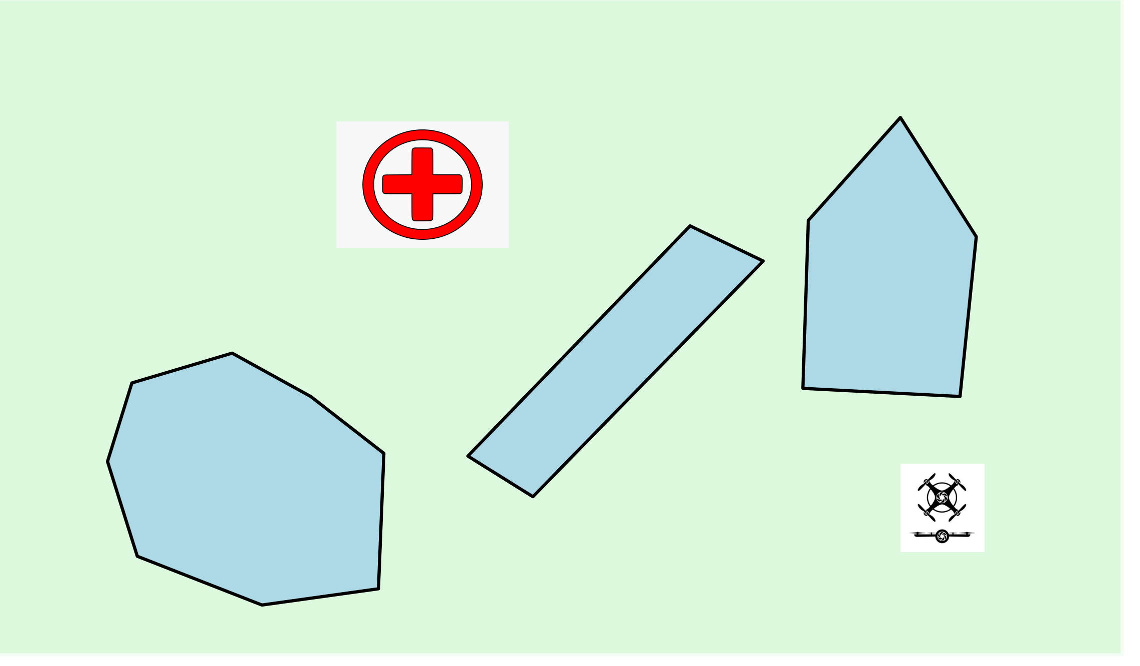


The solution is a *subdivision* of $\mathbb{R}^2$, called **Voronoi Diagram**.....
Lots of applications, e.g.: site planning, facility placement, nearest-neighbor finding, wireless networks ...

Same approach applies to find some serious services - Hospitals, Bank, Metro Station, Institions, etc.

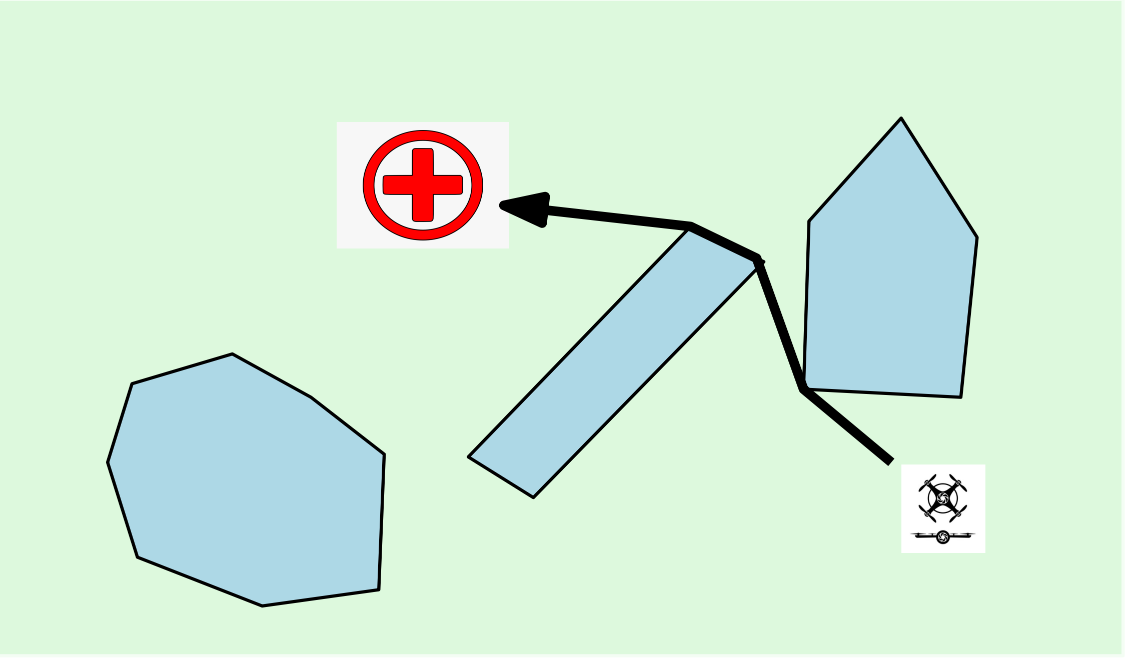
Problem 2

We want to send a robot to go to a hospital/ medical store. How can the robot reach the destination without passing through houses, park benches, and trees?



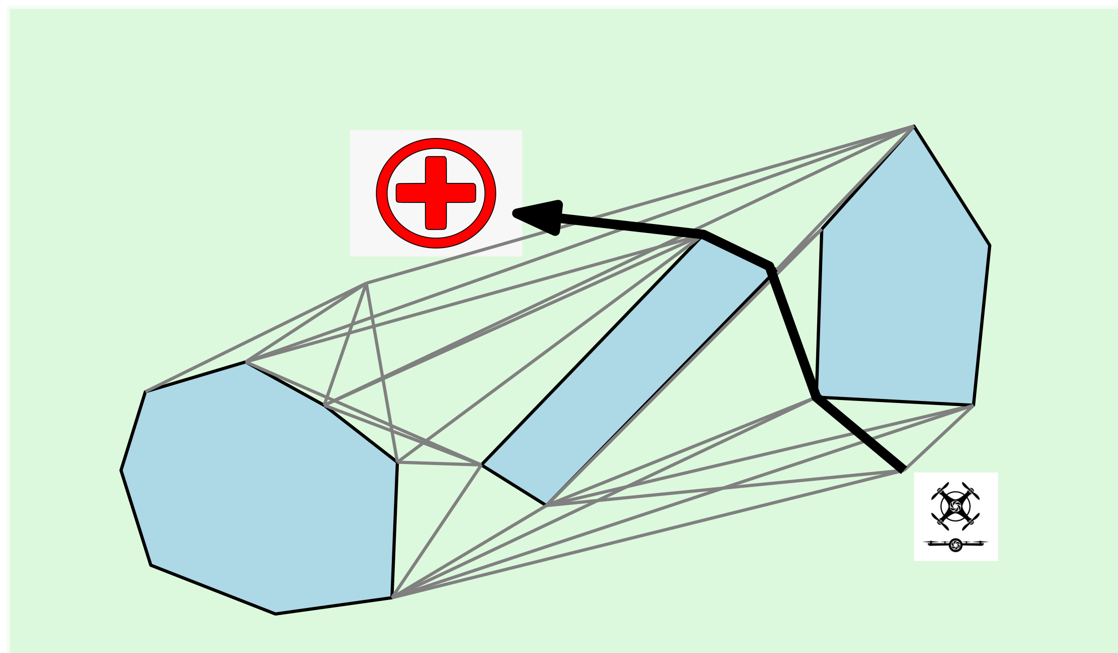
Problem 2

We want to send a robot to go to a hospital/ medical store. How can the robot reach the destination without passing through houses, park benches, and trees?



Problem 2

We want to send a robot to go to a hospital/ medical store. How can the robot reach the destination without passing through houses, park benches, and trees?

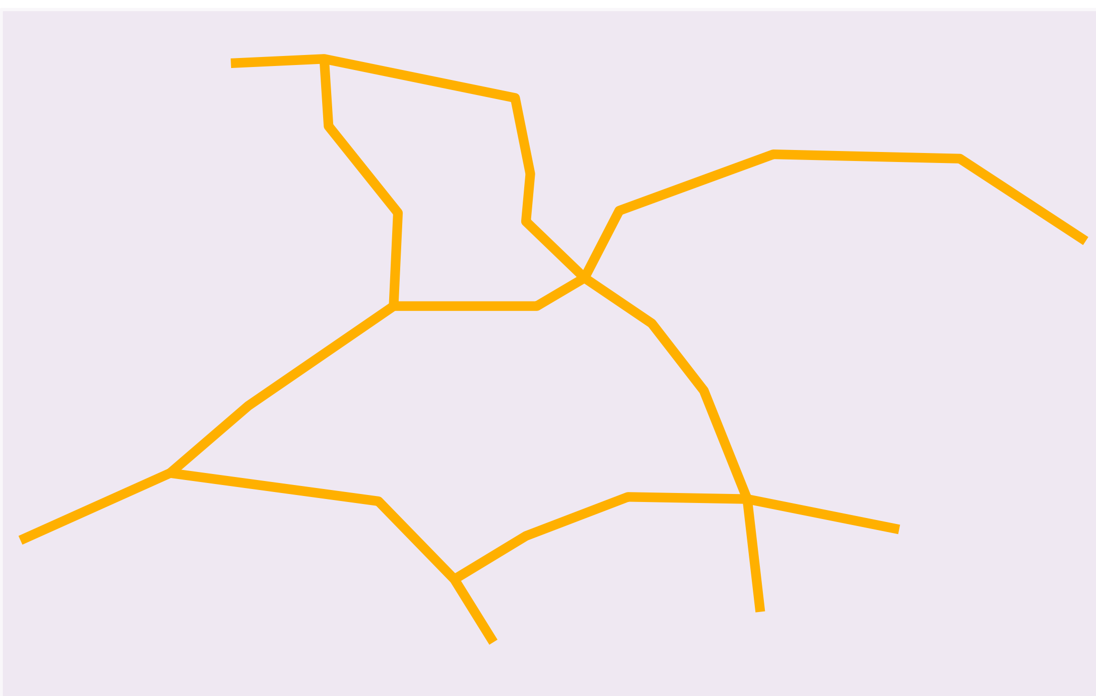


Motion planning problem in robotics:
Given a set of obstacles with a start and destination point,
find collision-free shortest route, e.g., using the **visibility graph.**

Problem 3

Maps in geographic information systems consist of several layers (e.g., roads, rivers, borders, etc.). When superimposing several layers, where are the intersection points?

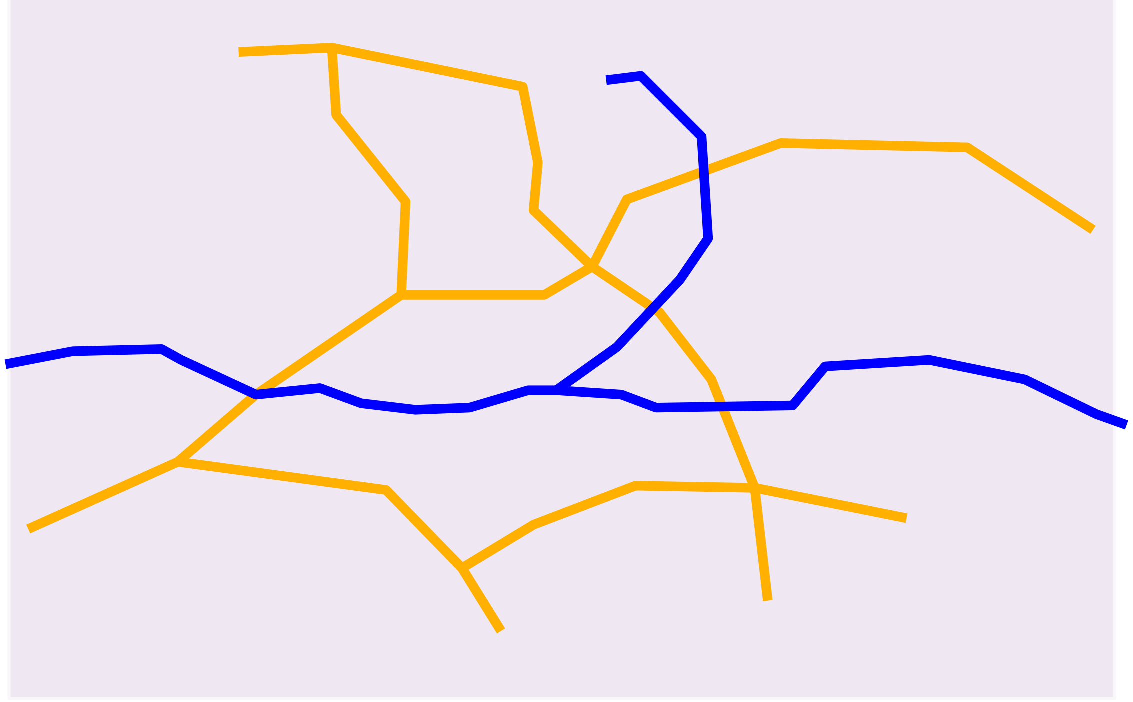
One example is to to view all roads and rivers as a set of line segments and ask for the river crossings. For these, you have to find all intersections between the two layers.



Problem 3

Maps in geographic information systems consist of several layers (e.g., roads, rivers, borders, etc.). When superimposing several layers, where are the intersection points?

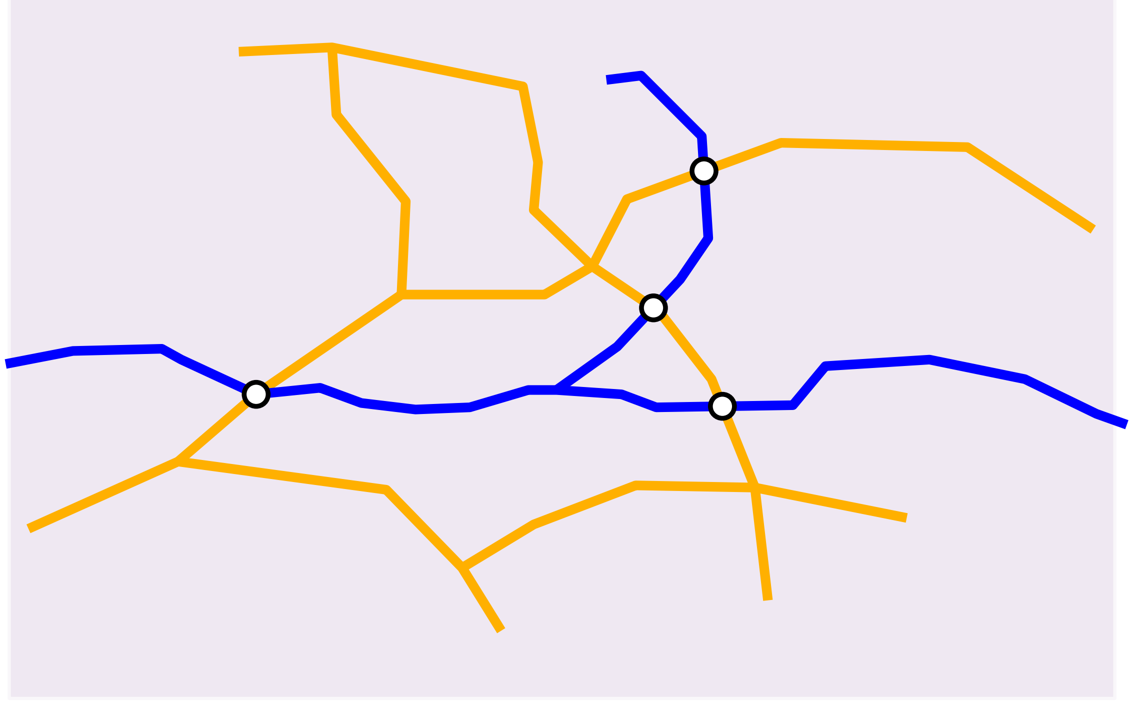
One example is to to view all roads and rivers as a set of line segments and ask for the river crossings. For these, you have to find all intersections between the two layers.



Problem 3

Maps in geographic information systems consist of several layers (e.g., roads, rivers, borders, etc.). When superimposing several layers, where are the intersection points?

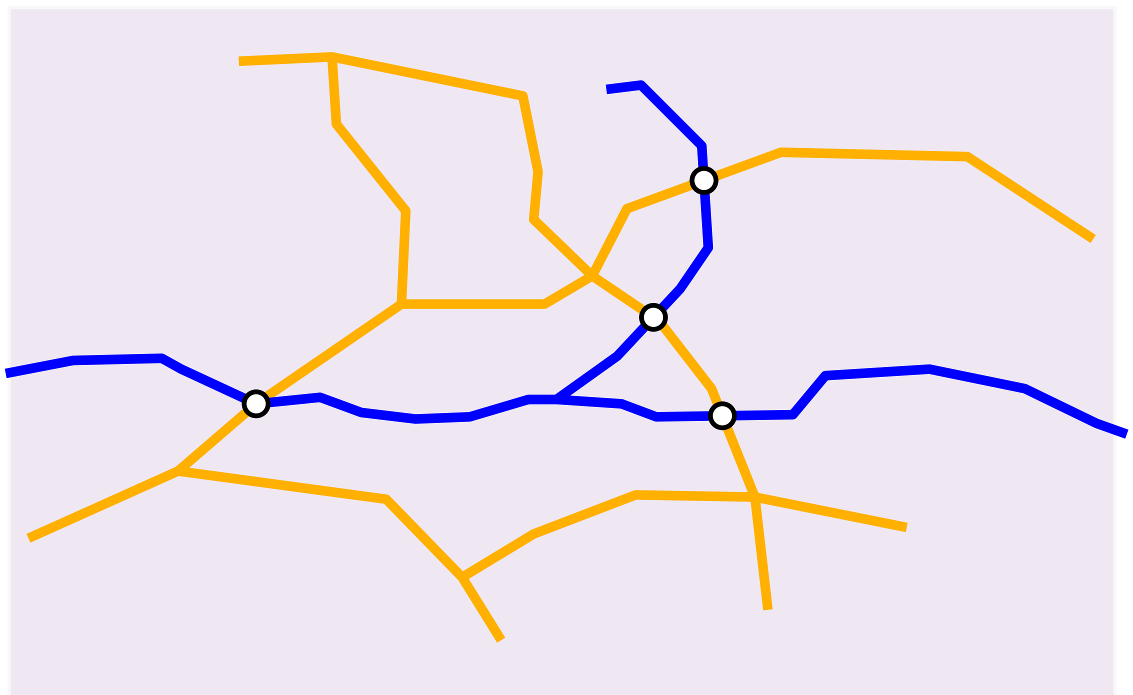
One example is to to view all roads and rivers as a set of line segments and ask for the river crossings. For these, you have to find all intersections between the two layers.



Problem 3

Maps in geographic information systems consist of several layers (e.g., roads, rivers, borders, etc.). When superimposing several layers, where are the intersection points?

One example is to to view all roads and rivers as a set of line segments and ask for the river crossings. For these, you have to find all intersections between the two layers.



Testing all edge pairs is slow.

Q: How can you quickly find all intersections?

Problem 4

Given a map and a query point q (e.g., a mouse click), determine the country containing q.

Problem 4

Given a map and a query point q (e.g., a mouse click), determine the country containing q.

We want to obtain efficient & robust data structures for answering point queries.

Problem 5

A navigation system should display the current map view.
How can we efficiently choose the data to display?

Problem 5

A navigation system should display the current map view.
How can we efficiently choose the data to display?

Problem 5

A navigation system should display the current map view.
How can we efficiently choose the data to display?

Evaluating each map feature is unrealistic.

→ We want efficient & robust data structures for answering such range queries

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%

Can we mix ... ?

q_1	15%	20%
q_2	25%	28%

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%

Can we mix ... ?

q_1	15%	20%
q_2	25%	28%

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%
S_3	40%	25%

Can we mix ... ?

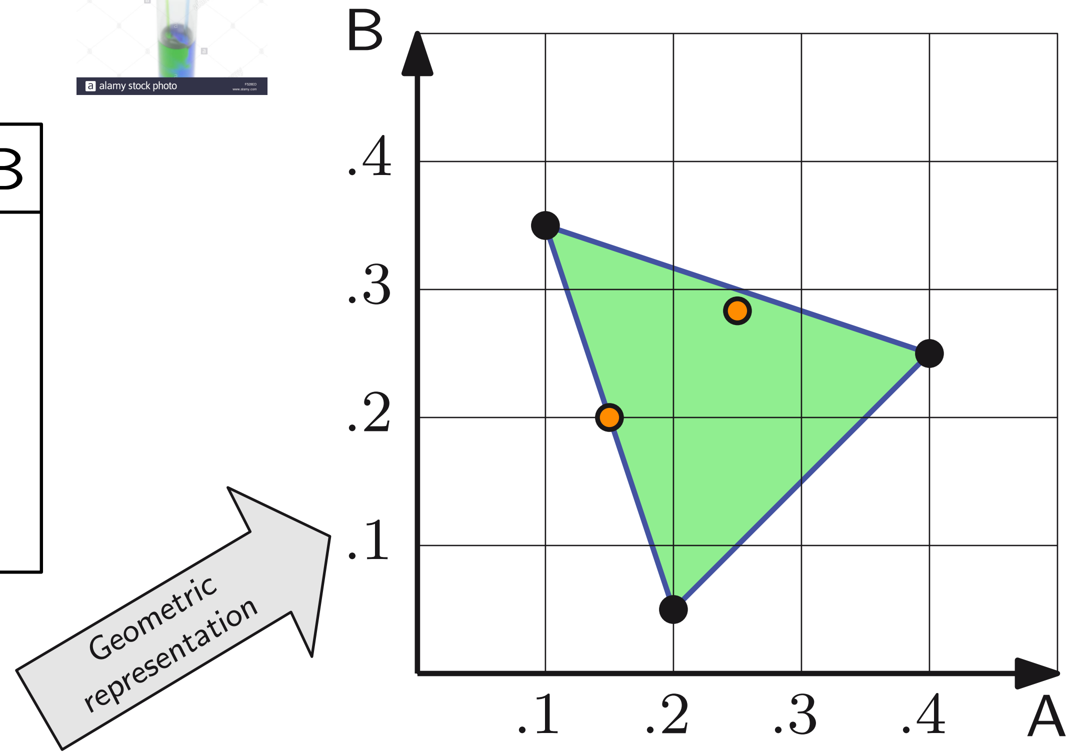
q_1	15%	20%
q_2	25%	28%

✓

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%
S_3	40%	25%



Can we mix ... ?

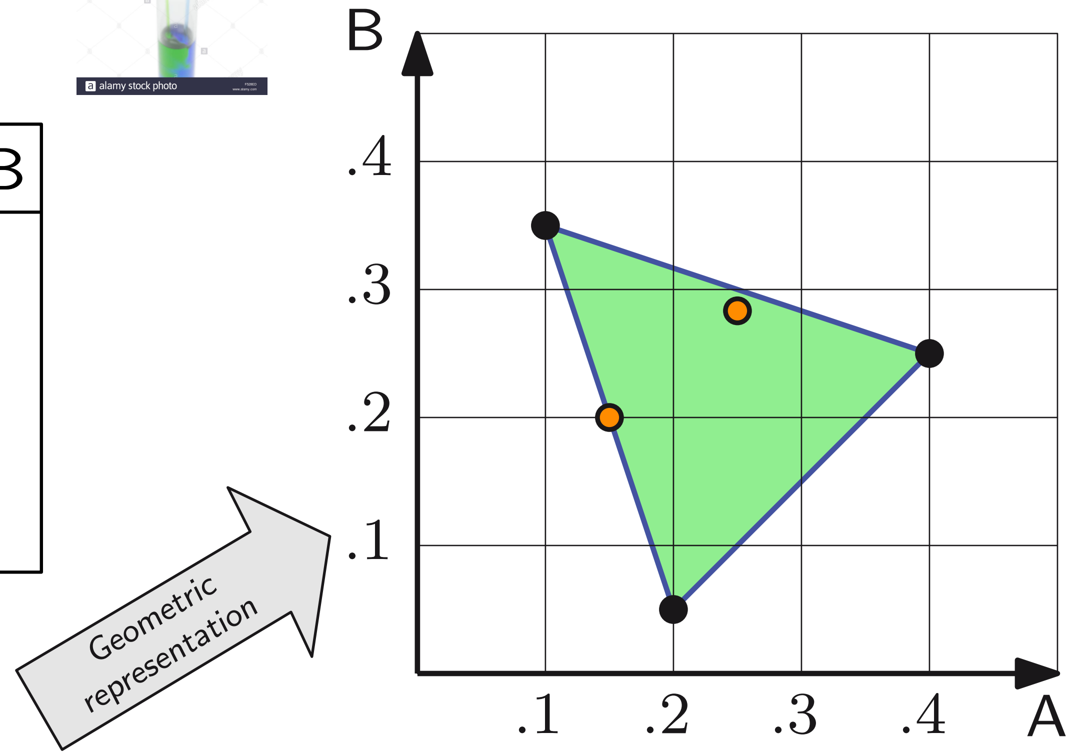
q_1	15%	20%
q_2	25%	28%

✓

Mixing Ratios

We are given -

Mixture	fraction A	fraction B
S_1	10%	35%
S_2	20%	5%
S_3	40%	25%



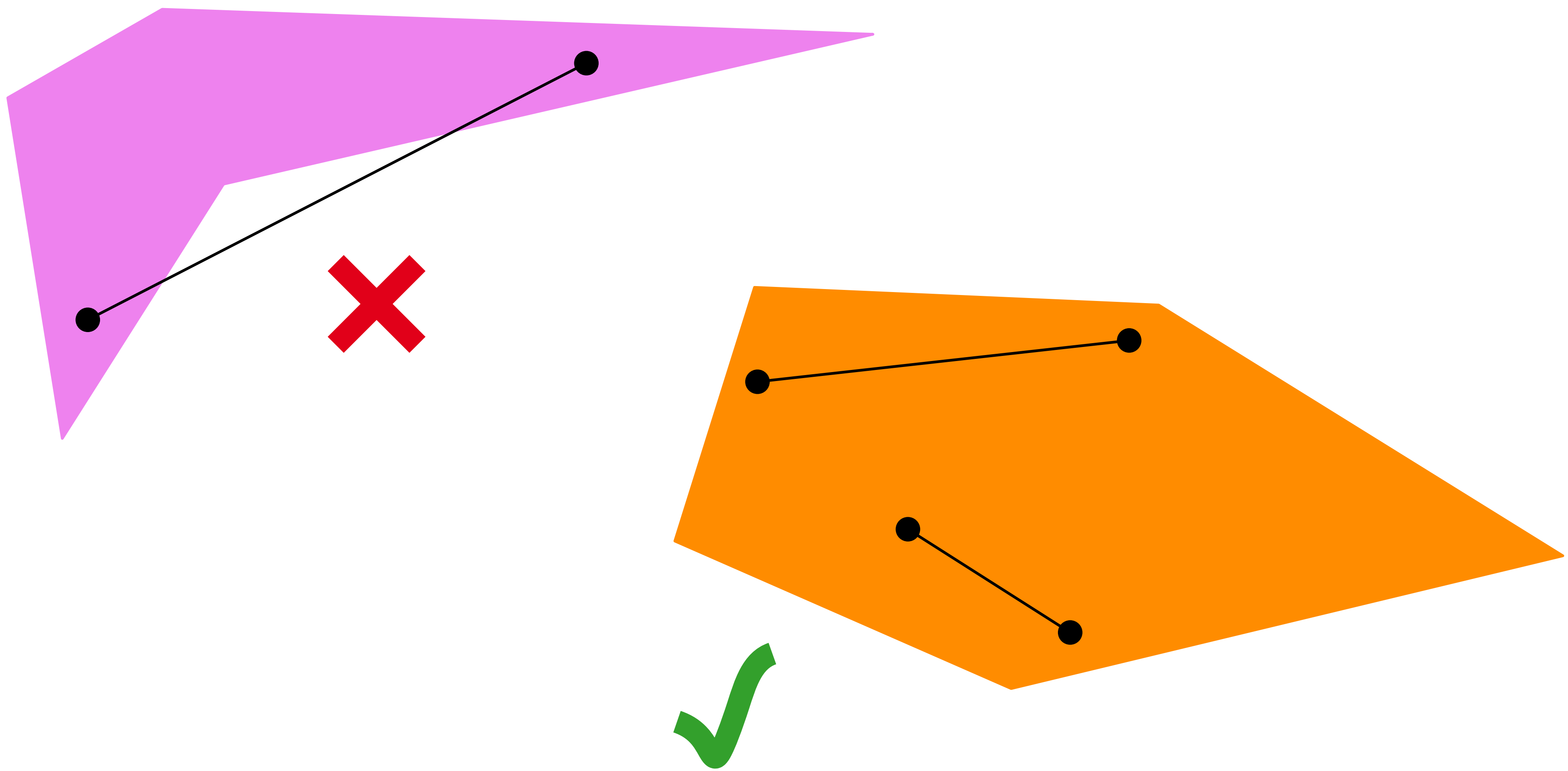
Can we mix ... ?

q_1	15%	20%
q_2	25%	28%

Observation. Given a set $S \subseteq \mathbb{R}^2$ mixtures, it is possible to make another mixture $q \in \mathbb{R}^2$ using $S \iff q \in$ ConvexHull $CH(S)$.
$q = \sum_i \lambda_i s_i$ with $\sum_i \lambda_i = 1$.

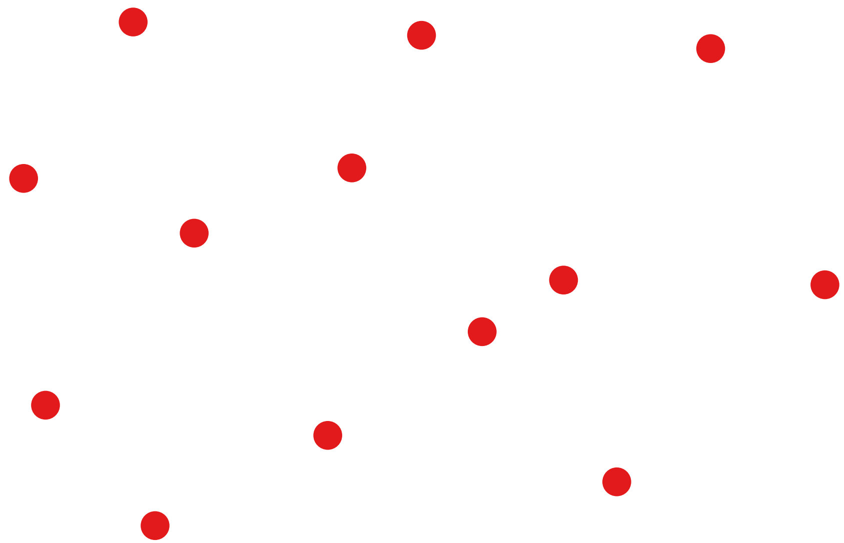
Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.



Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

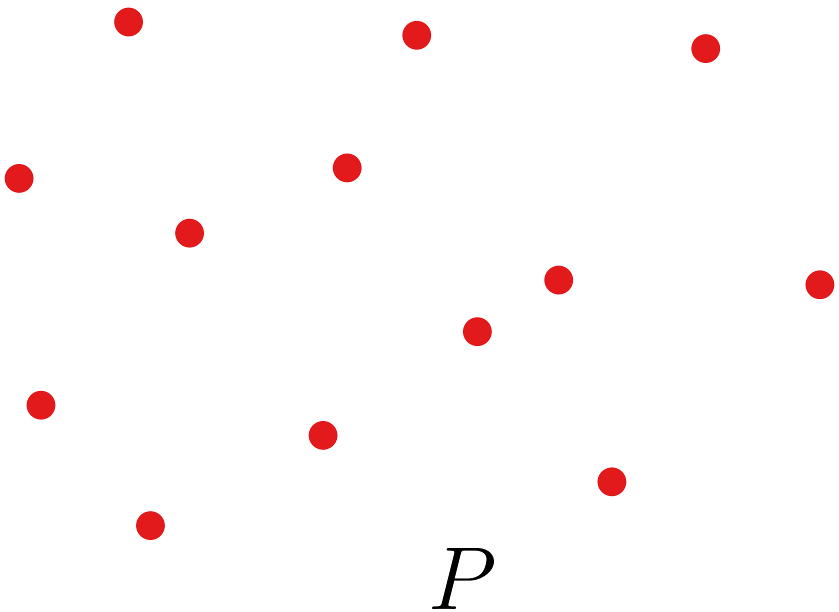


P

Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

How about some physics -

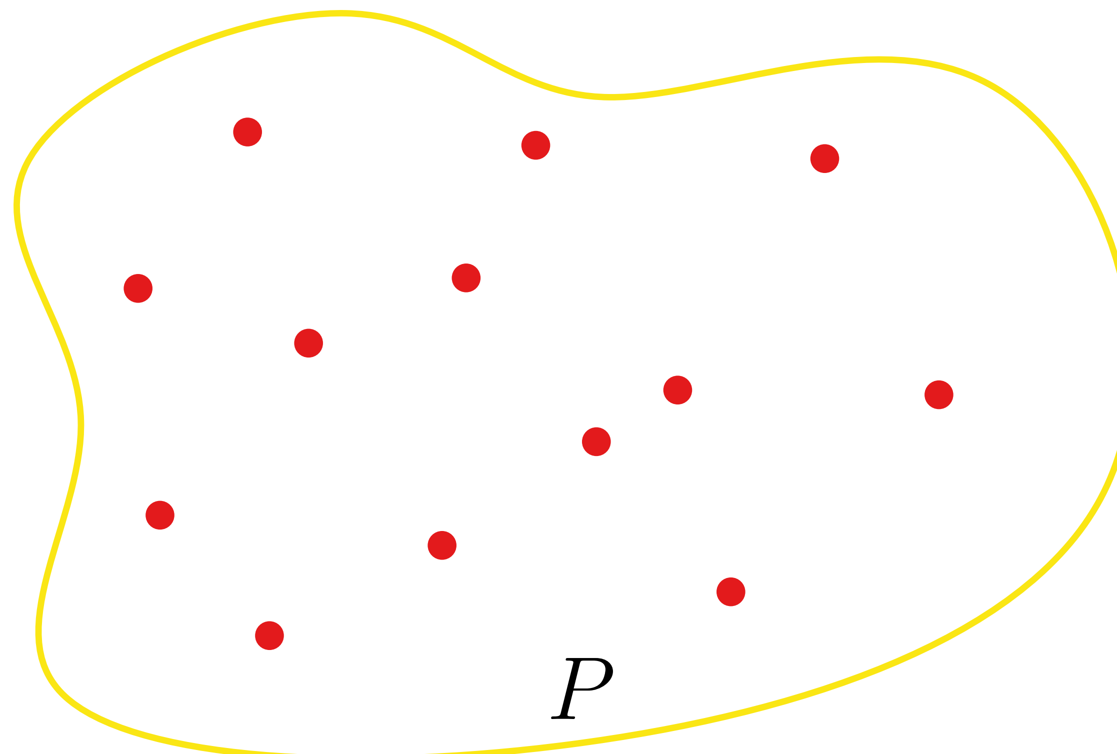


Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

How about some physics -

- put a large rubber band around all points

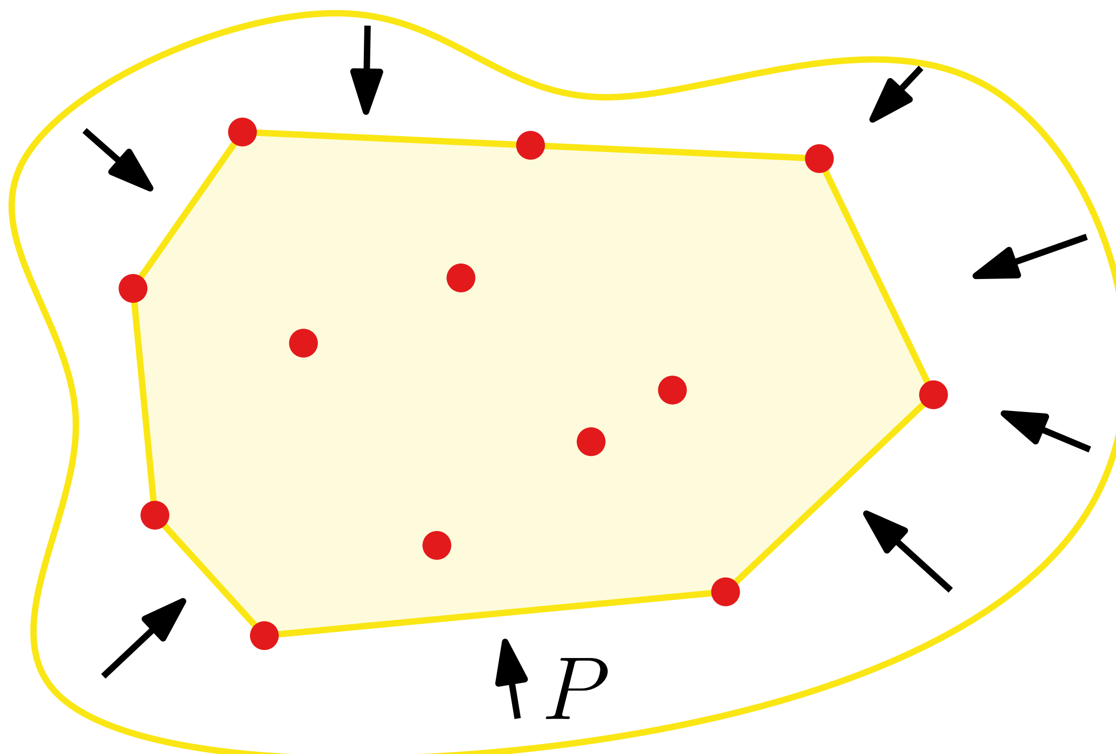


Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

How about some physics -

- put a large rubber band around all points
- and let it go!

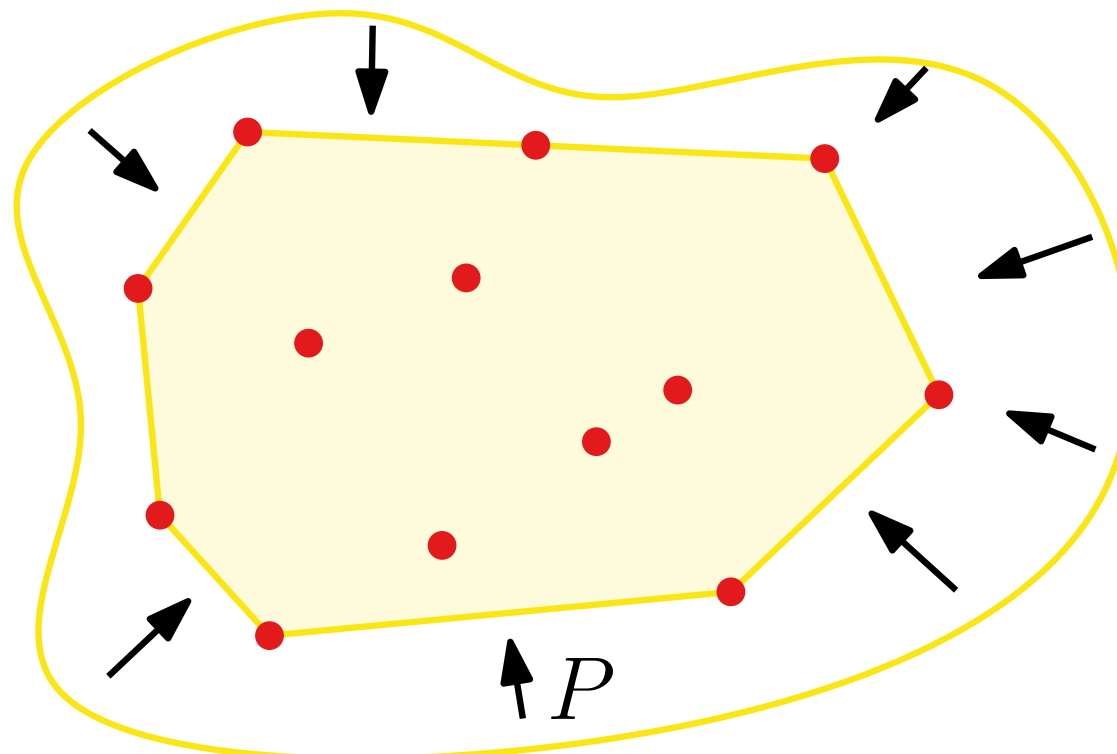


Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

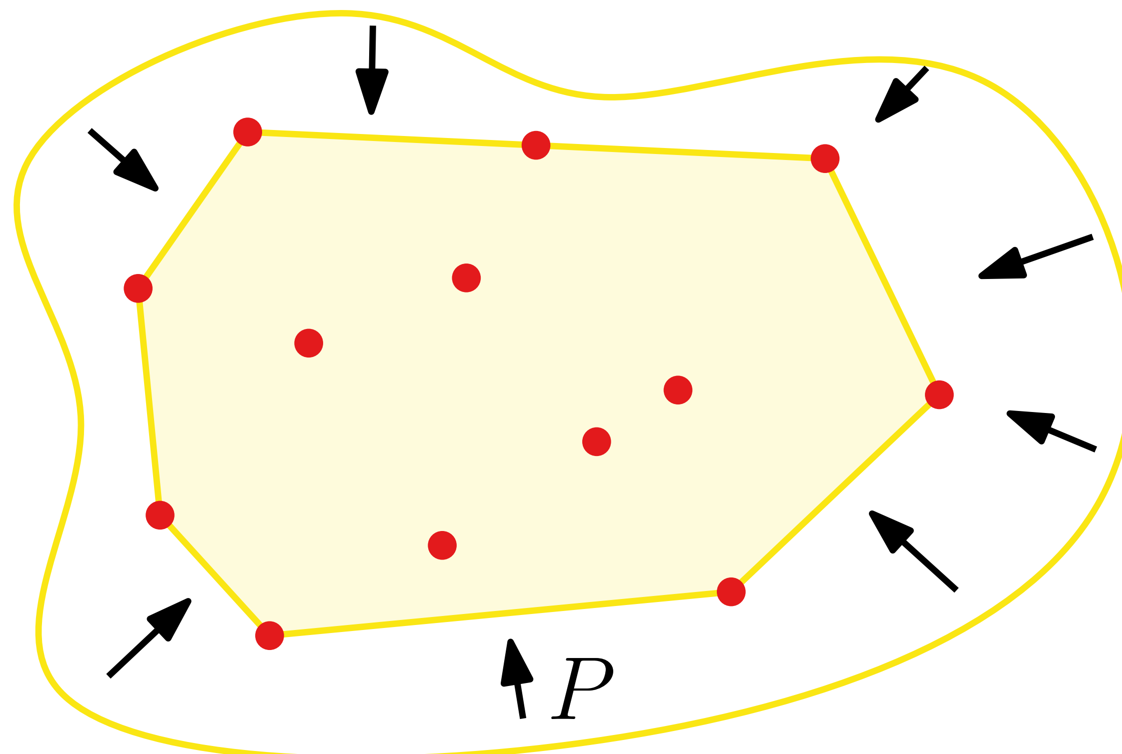
How about some physics -

- put a large rubber band around all points
- and let it go!
- unfortunately, does not help algorithmically



Definition of Convex Hull

Definition. A region $S \subseteq \mathbb{R}^2$ is called **convex**, if for any two points $p, q \in S$ the line segment $\overline{pq} \in S$. The **convex hull** $CH(S)$ of S is the smallest convex region containing S.



How about some physics -

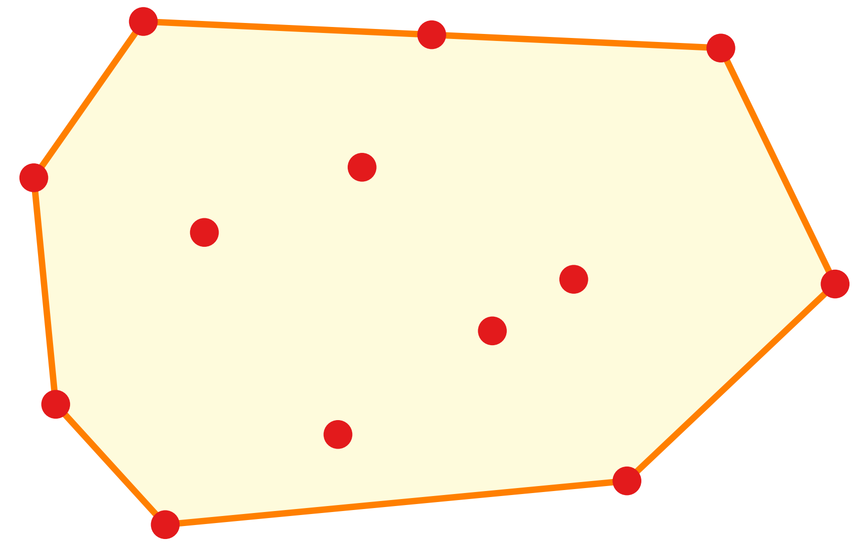
- put a large rubber band around all points
- and let it go!
- unfortunately, does not help algorithmically

Now, some Mathematics -

- define $CH(S) = \bigcap_{C \supseteq S :\, C \text{ convex}} C$
- does not help either.

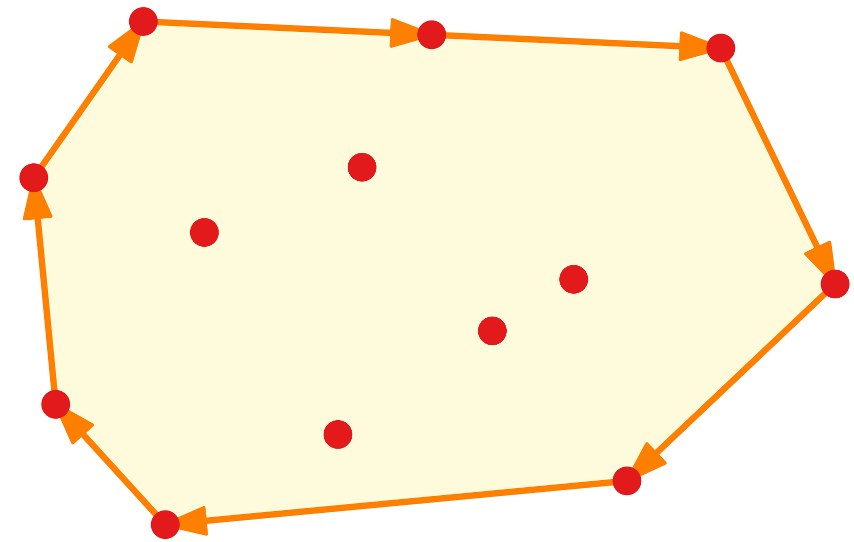
Algorithmic Approach

Lemma. For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are a subset of P.



Algorithmic Approach

Lemma. For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are a subset of P.

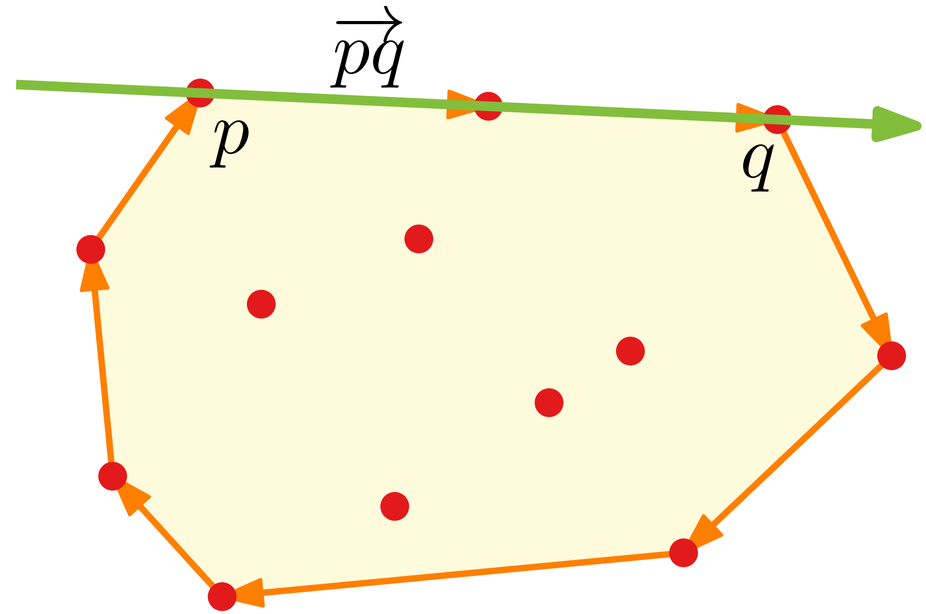


Input: A set of points $P = \{p_1, \ldots, p_n\}$

Output: List of vertices of $CH(P)$ in clockwise order

Algorithmic Approach

Lemma. For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are a subset of P.



Input: A set of points $P = \{p_1, \ldots, p_n\}$

Output: List of vertices of $CH(P)$ in clockwise order

Observation. (p, q) is an edge of $CH(P) \Leftrightarrow$ each point $r \in P \setminus \{p, q\}$

- strictly right of the oriented line $\overrightarrow{pq}$ or
- on the line segment $\overline{pq}$

Computing Convex Hull

ConvexHull(P)

$E \leftarrow \emptyset$
foreach $(p, q) \in P \times P$ with $p \neq q$ **do**
  valid $\leftarrow$ *true*
  foreach $r \in P$ **do**
    if not (r strictly right of $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**
      valid $\leftarrow$ *false*
  if *valid* **then**
    $E \leftarrow E \cup \{(p, q)\}$
construct sorted vertex list L of $CH(P)$ from E
return L

Computing Convex Hull

ConvexHull(P)

$E \leftarrow \emptyset$
foreach $(p, q) \in P \times P$ with $p \neq q$ **do**
  valid $\leftarrow$ *true*
  foreach $r \in P$ **do**
    if not (r strictly right of $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**
      valid $\leftarrow$ *false*
  if *valid* **then**
    $E \leftarrow E \cup \{(p, q)\}$
construct sorted vertex list L of $CH(P)$ from E
return L

Check all possible edges (p, q)

Computing Convex Hull

ConvexHull(P)

$E \leftarrow \emptyset$
foreach $(p, q) \in P \times P$ with $p \neq q$ **do**
  valid $\leftarrow$ *true*
  foreach $r \in P$ **do**
    if not (r strictly right of $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**
      valid $\leftarrow$ *false*
  if *valid* **then**
    $E \leftarrow E \cup \{(p, q)\}$
construct sorted vertex list L of $CH(P)$ from E
return L

Check all possible edges (p, q)

Test in $O(1)$ time with

$$\begin{vmatrix} x_r & y_r & 1 \\ x_p & y_p & 1 \\ x_q & y_q & 1 \end{vmatrix} < 0$$

$\rightarrow$ think (exercise-1)

Limits of Computation ...

What computers can not do ... ?

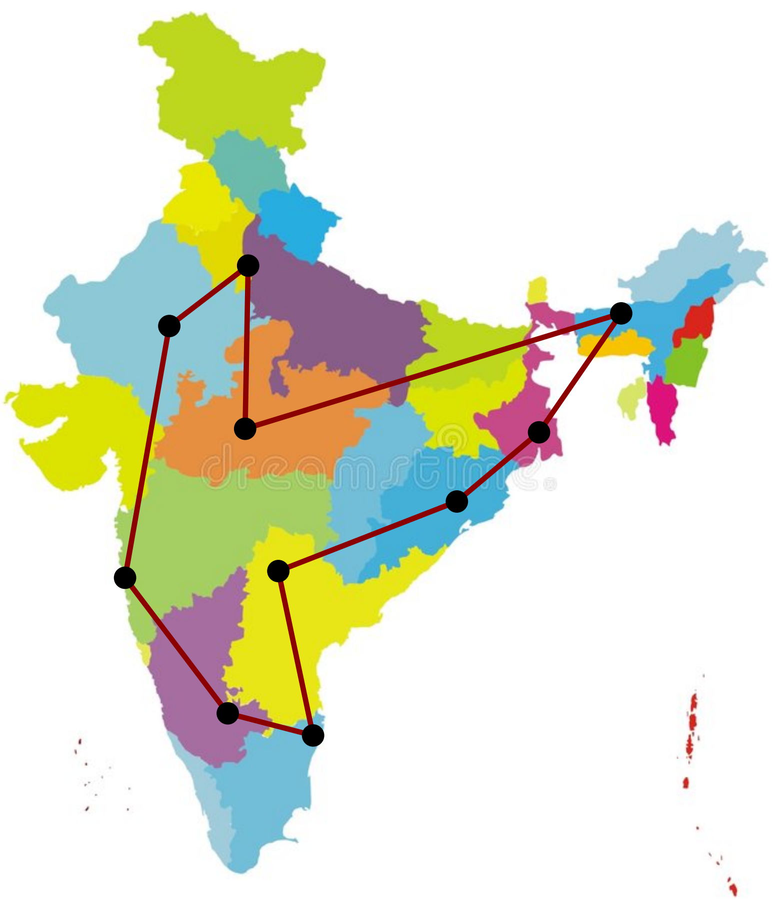
Limits of Computation ...

What computers can not do ... ?

Q: find shortest possible route that visits each city exactly once and returns to the origin city

Limits of Computation ...

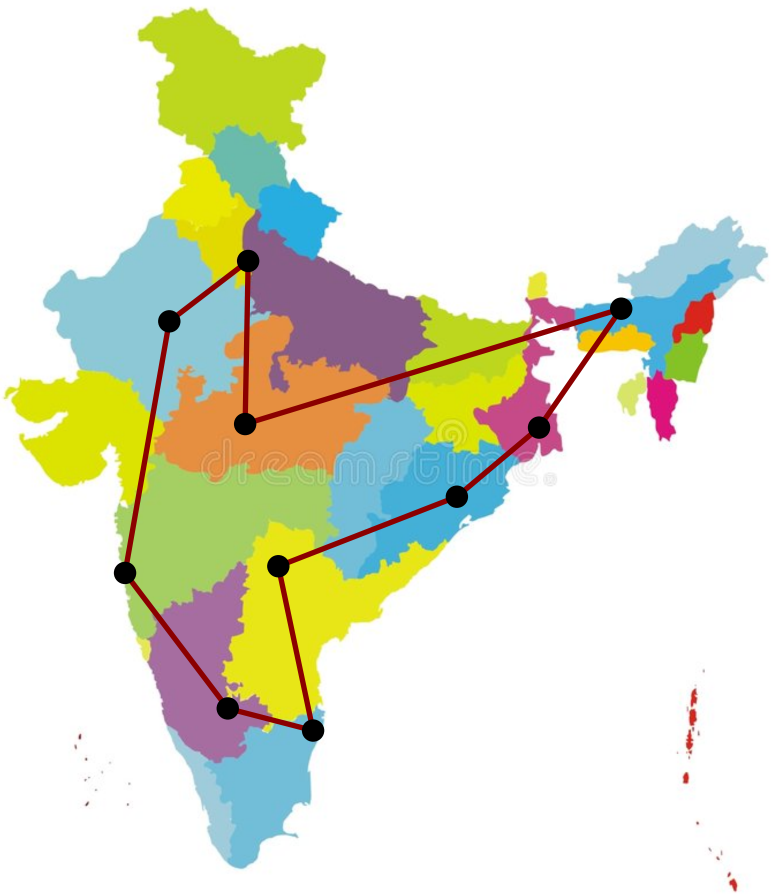
What computers can not do ... ?



Q: find shortest possible route that visits each city exactly once and returns to the origin city

Limits of Computation ...

What computers can not do ... ?



Q: find shortest possible route that visits each city exactly once and returns to the origin city

- main issue is Ordering!

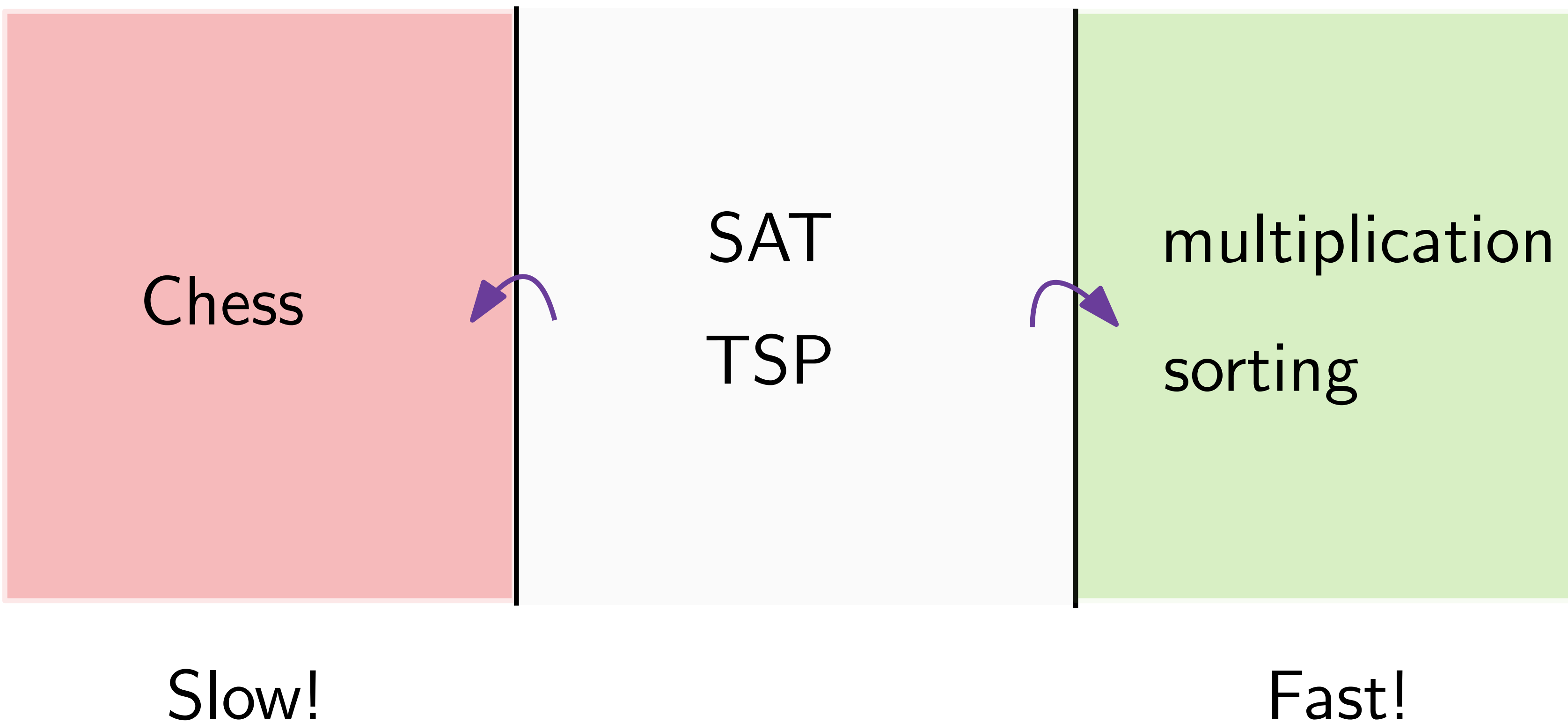
- even the fastest computers can not afford exponential computation.

- time and space are invaluable.

Limits of Computation ...

What computers can not do ... ?

Categories of problems



Complexity Classes ...

P $= \{$Problems that are solvable in polynomial time.$\}$

If the problem has size n, then it can be solved in $n^{O(1)}$ time.

Complexity Classes ...

P = {Problems that are solvable in polynomial time.}

If the problem has size n, then it can be solved in $n^{O(1)}$ time.

NP = {Decision problems solvable in nondeterministic polynomial time.}

Complexity Classes ...

P = {Problems that are solvable in polynomial time.}

If the problem has size n, then it can be solved in $n^{O(1)}$ time.

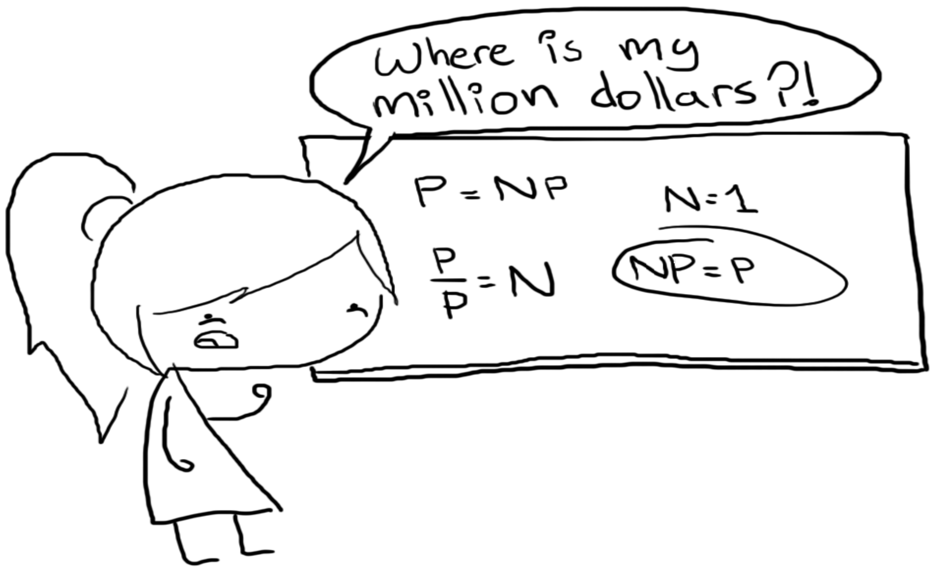
NP = {Decision problems solvable in nondeterministic polynomial time.}

Output is **YES** or **NO**.

In $O(1)$ time can "guess" among polynomial number of choices & if any guess leads to YES, then the nondeterministic algorithm will make that guess.

There is an asymmetry between YES and NO inputs.

The BIG problem !!!

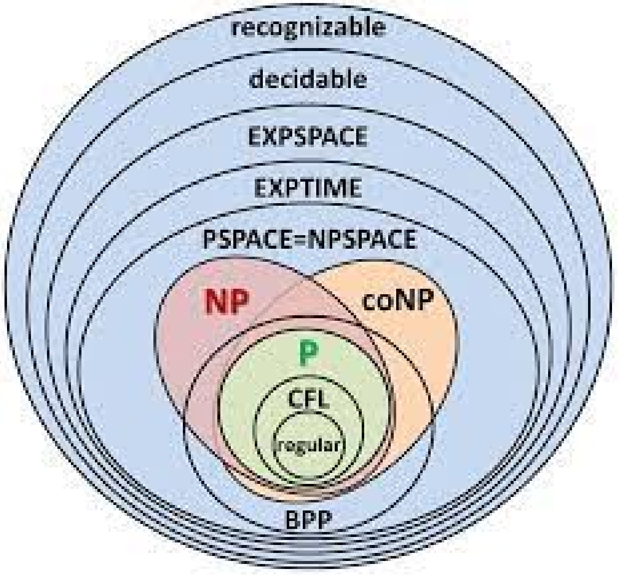


on a hilarious note!

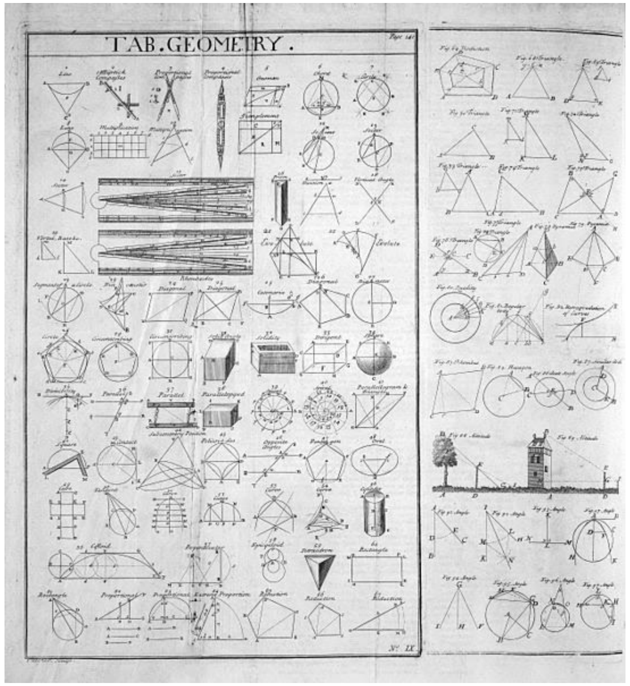
The BIG problem !!!

on a hilarious note!

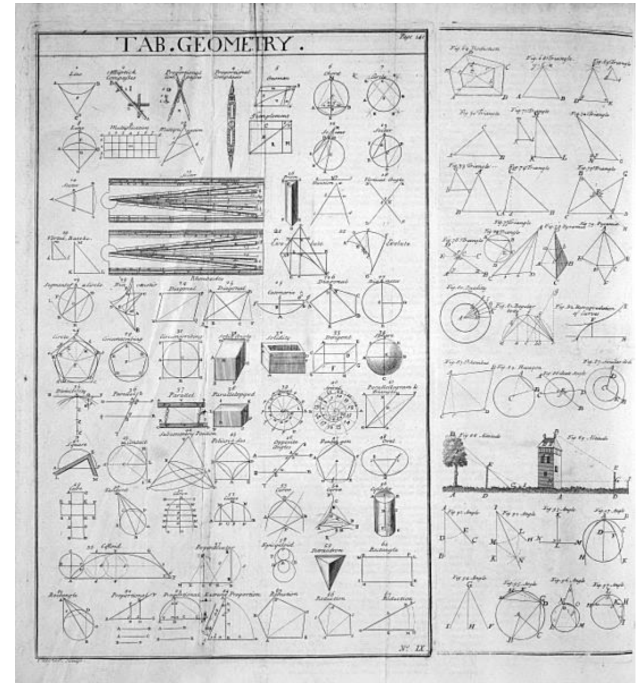
on a serious note!



Let's begin the learning together ...



Let's begin the learning together ...



Thank you!