Reductions

Asymptotic lower bound: $f(n)$ such that any solution for X has worst case $\Omega(f(n))$

Reduction: reduce Y to X by solving Y using an algorithm to solve X

reducing $\underbrace{\text{bipartite matching}}_{Y}$ to $\underbrace{\text{maximum flow}}_{X}$

an upper bound for X is an upper bound for Y

a lower bound for Y is a lower bound for X

bipartite graph G

↓ transform input — $O(m)$

capacity graph G'

↓ run algorithm for X: Ford-Fulkerson — $O(Cm) = O(nm)$

maximum flow f

↓ transform output — $O(m)$

maximum matching M

total: $O(nm)$

Asymptotic lower bound: $f(n)$ such that any solution for X has worst case $\Omega(f(n))$

Reduction: reduce Y to X by solving Y using an algorithm to solve X

reducing $\underbrace{\text{bipartite matching}}_{Y}$ to $\underbrace{\text{maximum flow}}_{X}$

an upper bound for X is an upper bound for Y

a lower bound for Y is a lower bound for X

bipartite graph G

transform input $\downarrow$ $O(m)$

capacity graph G'

run algorithm for X $\downarrow$ Ford-Fulkerson $O(Cm) = O(nm)$

maximum flow f

transform output $\downarrow$ $O(m)$

maximum matching M

total: $O(nm)$

Asymptotic lower bound: $f(n)$ such that any solution for X has worst case $\Omega(f(n))$

Reductions

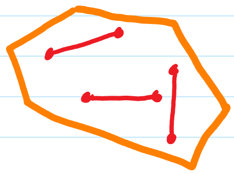
Asymtotic lower bound: $f(n)$ such that any solution for X has worst case $\Omega(f(n))$

Reduction: reduce Y to X by solving Y using an algorithm to solve X

2-D Convex Hull

2-D Convex Hull: given set of points, find subset that defines a convex polygon that contains them all (7 from bottom)

lines between interior points stay inside

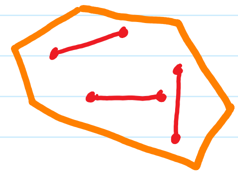


non-convex polygons

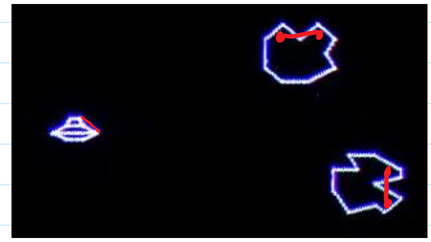
2-D Convex Hull

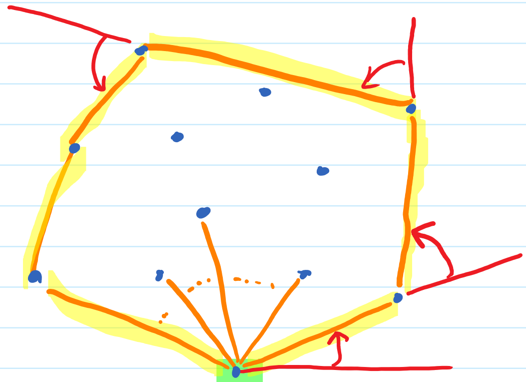
2-D Convex Hull: given set of points, find subset that defines a **convex** polygon that contains them all (↺ from bottom)

lines between interior points stay inside



non-convex polygons

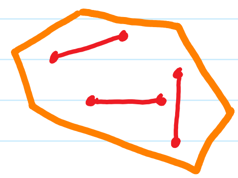




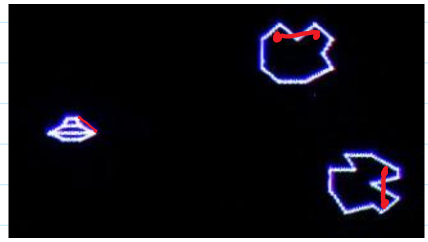
2-D Convex Hull

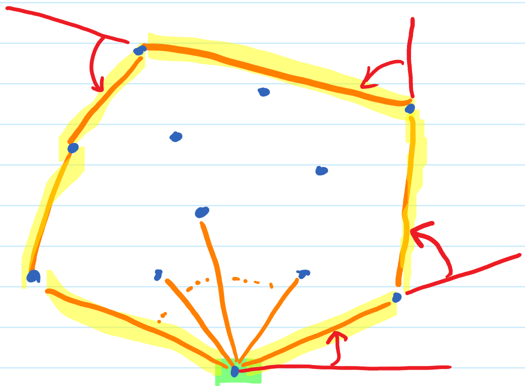
2-D Convex Hull: given set of points, find subset that defines a <u>convex</u> polygon that contains them all (↺ from bottom)

lines between interior points stay inside



non-convex polygons





Jarvis March: $O(n^2)$
(gift wrapping)

Graham Scan: $O(n \log n)$

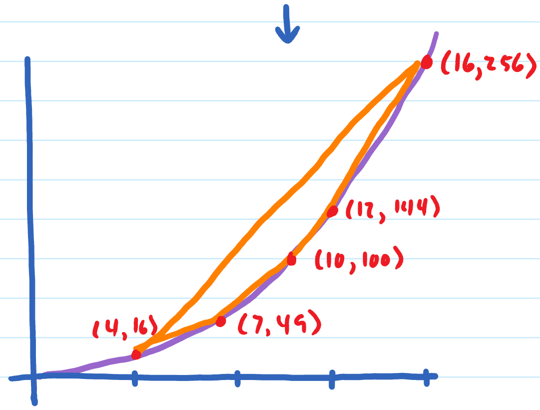
Convex Hull Sort

list of positive integers

```
CONVEX-HULL-SORT(A)
  P ← empty list
  for each x ∈ A
      add (x, x²) to P

  H ← CONVEX-HULL(P)

  A ← empty list
  for each (x,y) ∈ P
      add x to A

  return A
```

[16, 12, 4, 7, 10]

↓



↓

[(4,16), (7,49), (10,100), (12,144), (16,256)]

↓

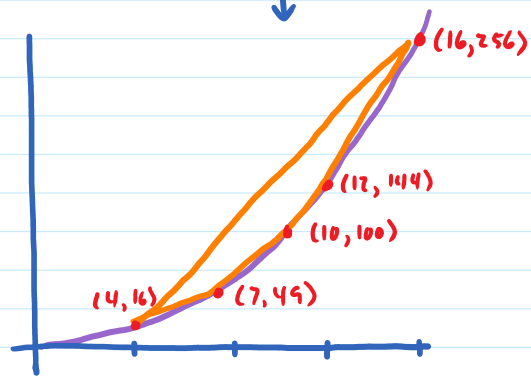
[4, 7, 10, 12, 16]

Convex Hull Sort

```
CONVEX-HULL-SORT(A)          list of positive integers
  P ← empty list
  for each x ∈ A                         O(n)
      add (x, x²) to P

  H ← CONVEX-HULL(P)                    ???

  A ← empty list
  for each (x,y) ∈ H                     O(n)
      add x to A

  return A
```

[16, 12, 4, 7, 10]

↓



↓

[(4,16), (7,49), (10, 100), (12,144), (16,256)]

↓

[4, 7, 10, 12, 16]

Suppose ??? is $O(n)$.

Then CONVEX-HULL-SORT runs in $O(n)$ time ⇏⇐

CONVEX-HULL worst case is $\Omega(n \log n)$ [lower bound for sorting is lower bound for CH]

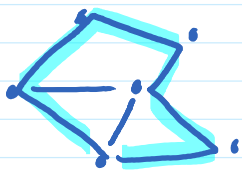
HC and TSP

visits each vertex exactly once

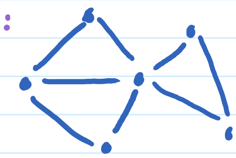
Hamiltonian Cycle: given undirected G, return Hamiltonian cycle, or report that none exists

given undirected G, determine if G has a Hamiltonian cycle

has HC:



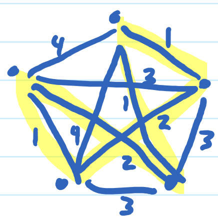
no HC:



Decision problem: output is Y or N

a Hamiltonian cycle

Travelling Salesperson: given fully connected, undirected, weighted G, find tour of lowest total weight



given fully connected, undirected, weighted G, and bound k, determine if G has a tour of total weight $\le k$

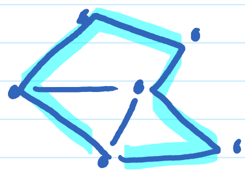
HC and TSP

visits each vertex exactly once

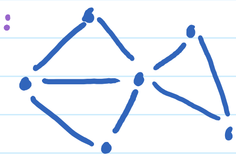
Hamiltonian Cycle: given undirected G, return Hamiltonian cycle, or report that none exists

given undirected G, determine if G has a Hamiltonian cycle

has HC:



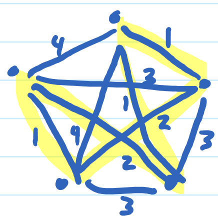
no HC:



Decision problem: output is Y or N

a Hamiltonian cycle

Travelling Salesperson: given fully connected, undirected, weighted G, find tour of lowest total weight



given fully connected, undirected, weighted G, and bound k, determine if G has a tour of total weight $\leq k$

Reducing HC to TSP

- solve HC using a solution to TSP

Given undirected G, construct fully connected, undirected, weighted G' by

1) copying the vertices of G
2) adding edges between all pairs of vertices
3) setting $w(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ exists in } G \\ 2 & \text{otherwise} \end{cases}$
4) choosing $k = \#$ of vertices

G has HC $\rightarrow$ G' has a tour of total weight $\leq k = n$

Suppose G has HC. Call it $v_0, v_1, \ldots, v_n = v_0$. Since that is a path in G, the edge (v_i, v_{i+1}) exists in G and so has $w(v_i, v_{i+1}) = 1$ by construction of G'. The path is also a tour of G' with total weight $n = k$.

G' has a tour of total weight $\leq k = n$ $\rightarrow$ G has a Hamiltonian Cycle

Suppose G' has a tour of total weight $\leq k = n$. It has n edges, each with weight 1 or 2, so all the edges must have weight 1 since otherwise the total weight is $> n$. By construction of G', all edges in the tour are edges in G. G' and G have the same vertex set, so the tour is also a Hamiltonian Cycle in G.

Reducing HC to TSP - solve HC using a solution to TSP

Given undirected G, construct fully connected, undirected, weighted G' by

1) copying the vertices of G
2) adding edges between all pairs of vertices
3) setting $w(u,v) = \begin{cases} 1 \text{ if } (u,v) \text{ exists in } G \\ 2 \text{ otherwise} \end{cases}$
4) choosing $k = \#$ of vertices

HC (G)

poly 1) construct G', choose k

2) answer ← TSP(G', k)

poly 3) return answer

poly # calls to TSP

G has HC → G' has a tour of total weight $\le k = n$

G has a Hamiltonian cycle if and only if G' has a tour of weight $\le n$

G' has a tour of total weight $\le k = n$ → G has a Hamiltonian Cycle

Polynomial-Time Reductions

polynomial-time reducible to

$Y \le_p X$ means there is an algorithm for Y that runs in polynomial time, aside from a polynomial number of calls to an algorithm that solves X

$HC \le_p TSP$ $HC \in P$??? $TSP \in P$???

P = set of decision problems solvable in polynomial time

$TSP \in P \to HC \in P$: Suppose $TSP \in P$. Then $HC(G)$ is a poly-time algorithm for HC

poly 1) construct G', choose k

poly 2) answer $\leftarrow TSP(G', k)$

poly 3) return answer

$HC \notin P \to TSP \notin P$ (contrapositive)

Suppose $Y \le_p X$ and $X \in P$. Then $Y \in P$ (use the algorithm from the def. of $\le_p$)

Suppose $Y \le_p X$ and $Y \notin P$. Then $X \notin P$ (contrapositive)

Polynomial-Time Reductions

polynomial-time reducible to

$Y \leq_p X$ means there is an algorithm for Y that runs in polynomial time, aside from a polynomial number of calls to an algorithm that solves X

$HC \leq_p TSP$

$HC \in P$???
$TSP \in P$???

P = set of decision problems solvable in polynomial time

$TSP \in P \rightarrow HC \in P$: Suppose $TSP \in P$. Then $HC(G)$ is a poly-time algorithm for HC

poly 1) construct G', choose k
poly 2) answer $\leftarrow TSP(G', k)$
poly 3) return answer

$HC \notin P \rightarrow TSP \notin P$ (contrapositive)

Suppose $Y \leq_p X$ and $X \in P$. Then $Y \in P$ (use the algorithm from the def. of $\leq_p$)

Suppose $Y \leq_p X$ and $Y \notin P$. Then $X \notin P$ (contrapositive)

Longest Path

Given a graph G, what is a longest simple path in G?