Scientific Computing

Feb 3, 2025

Announcements

- → HW 2 assigned, due in two weeks (Mon, Feb. 17)
- → Mon, Feb. 17, no in-person lecture and no office hours
- → Wed, March 5, midterm exam, in person portion (in class)
- → Fri, March 7, no class, extra office hours for take-home portion (time TBD)

Today

- → Greedy Algorithms
- → Unix command line

Office Hours:

Mon + Fri

9:30am - 10:30am

Cudahy 307

Problem #4 - Knapsack Problem

You have n items that each have a value v_i and a weight w_i. You have a knapsack that can carry a total weight of C. What is the highest value of items you can carry in your knapsack?

capacity

Problem #4 - Knapsack Problem

You have n items that each have a value v_i and a weight w_i. You have a knapsack that can carry a total weight of C. What is the highest value of items you can carry in your knapsack?

Ex:

item	weight	value
1	8	13
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

Capacity = 10

Possible solutions

* Items 3 and 4
  weight: $5+5 \le 10$ ✓
  value: $10+10 = 20$.
  optimal

* Items 3, 2, and 5
  weights: $5+3+2 = 10$ ✓
  value: $10+7+1 = 18$

Greedy possibilities:

* Always take lightest item (value = 10 in the example above)
* Always take most valuable (value = 14)
* Take the most value-dense: $\frac{\text{value}}{\text{weight}}$ (value = 20)

Are any of these guaranteed to be always optimal?

Ex:

item	weight	value	
1	8	13	1.625
2	3	7	2.33
3	5	10	2
4	5	10	2
5	2	1	0.5
6	2	1	0.5
7	2	1	0.5

Capacity = 10

Lightest:
5, 6, 7, 2
w = 9
v = 10

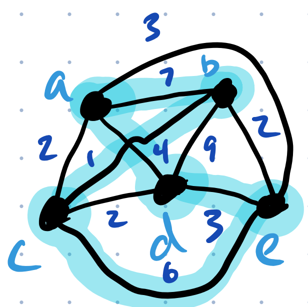
Valuable:
1, 5
w = 10
v = 14

Density:
2, 3, 5
w = 10
v = 18

Problem #5 - Traveling Salesman Problem (TSP)

There are n cities that a salesman needs to visit, and return. What is the shortest route that visits each city exactly once and returns back to the starting place?

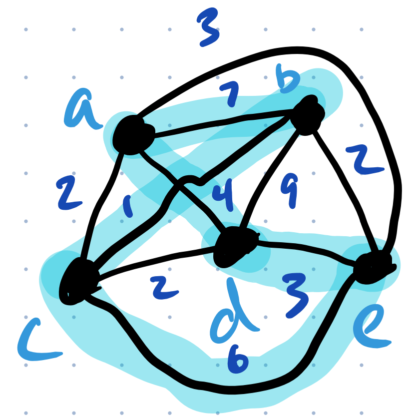
More formally: Consider a weighted graph G. Which ordering of the vertices gives you the smallest sum of edge weights?



$e \to c \to b \to a \to d \to e$

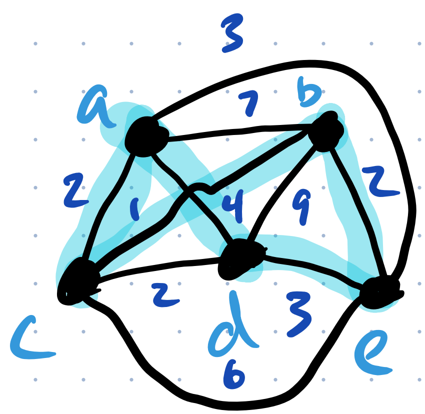
$a \to d \to e \to c \to b \to a$

$4 + 3 + 6 + 1 + 7 = 21$



$a \to d \to e \to c \to b \to a$

$4 + 3 + 6 + 1 + 7 = 21$

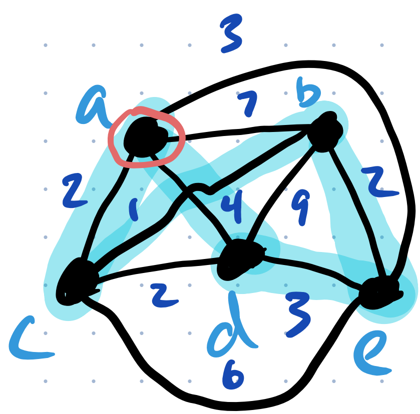


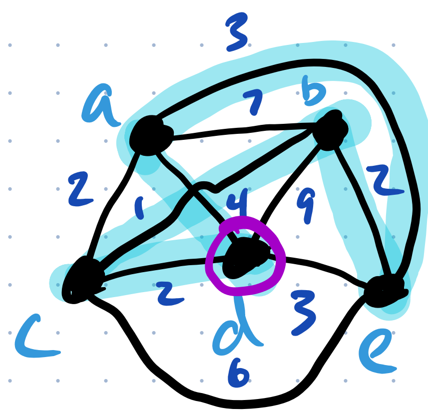
$a \to c \to b \to e \to d \to a$

$2 + 1 + 2 + 3 + 4 = 12$

Greedy algo: * pick a start vertex v_1.

* pick v_2 to be the closest vertex to v_1
* pick v_3 to be the closest unused vertex to v_2
* repeat until last vertex is picked then go back to v_1.





a → c → b → e → d → a

2 + 1 + 2 + 3 + 4 = 12

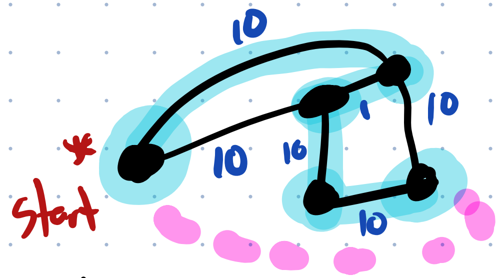
d → c → b → e → a → d

2 + 1 + 2 + 3 + 4

= 12

Notes: - only works if it's possible to go from any city to any other city, otherwise you might get stuck

Ex:



- does okay, but usually ends up picking some dumb edges (demo in a minute)

- brute force: $O((n-1)!)$
  (recall $k! = k \cdot (k-1) \cdot (k-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$)

$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$
$= 120$

$n! \approx n^n$

- dynamic programming: $O(n^2 \cdot 2^n)$
  still exponential!

This is just a really hard problem, but <u>very</u> intensely studied. We will use it as an example frequently when we learn "metaheuristic methods" later.

[Demos]