Approximation Algorithms

Sam McCauley

May 6, 2024

Welcome Back!

- Assignment 8 Wednesday
- SCS today (if you don’t have a computer that’s OK, but please fill it out!)
- Questions?

Final Exam: Updates

- On the final I will drop each student’s lowest-scoring question
- (Idea: less variance; some extra points; less time pressure; OK to be less comfortable on one topic)
- You can skip a question if you want. I’d recommend at least going for partial credit on all questions however

Let’s Make a Deal

- Approximation algorithms will not be on the final
- Nor will the “fun” NP-hard problems we talk about
- Stay with me while we have a low-key conversation about them
- Still good practice for final:
  - We’ll look at greedy approximation algorithms
  - We’ll talk about some counterexamples

Today

- Subset-Sum $\leq_P$ Knapsack
- Brief overview of other NP-hard problems
- Approximation algorithms
- End early; Course summary and SCS forms

Subset-Sum $\leq_P$ Knapsack

Showing that Knapsack is NP-hard

- Subset sum looks a little bit like knapsack (we’ll go over on the next slide)
- We couldn’t find a polynomial-time algorithm for knapsack and I claimed there wasn’t one
- Can we prove it?

(Recall) Knapsack

- You are packing a bag, with a weight capacity C
- You have a collection of items to put in your bag
- Each item i has a weight w_i and a value v_i (both nonnegative integers)
- Choose a subset of items with *total weight* at most C
- Goal: maximize the *total value* of the items you pack
- Goal (decision version): can we pack items with value at least V?

Subset-Sum $\leq_P$ Knapsack

- Prove Knapsack NP-hard
- Why is Knapsack in NP?
- To show the above: given an instance (S, T) of subset sum, want to create an instance of Knapsack such that we can pack items with total value $\geq V$ in the knapsack if and only if (S, T) has a subset sum

Comparing the Problems

Subset Sum:

- given: set of integers S
- goal: find a subset $S' \subseteq S$
- requirement: $\sum_{s \in S'} s = T$ (the elements of s sum to T)

Knapsack

- given: n items, each with a weight w_i and value v_i; capacity C; target value V
- goal: find a set of items with total value at least V
- requirement: the set of items have total weight at most C

Subset sum to Knapsack

Start with $S = s_1, \ldots, s_n$ and T.

In Knapsack, we need total value *at least* V and total weight *at most* C. In subset sum, we needed integers with total *exactly* T

- Idea: if we have $V = C = T$ then $\geq V, \leq C$ means $= T$
- We create a Knapsack instance with $V = C = T$
- For each item i in our knapsack, let $w_i = v_i = s_i$.
- Now: need to prove correctness.

Correctness Proof

If (S, T) has a subset sum, then our knapsack instance has a solution.

- Let S' be the solution to S, T; so $\sum_{s \in S'} s = T$.
- For each $s_i \in S'$, add item i to our Knapsack instance
- Total weight: $T = C$
- Total value: $T = V$
- So the items we have selected have total value at least V, and total weight at most C. $\square$

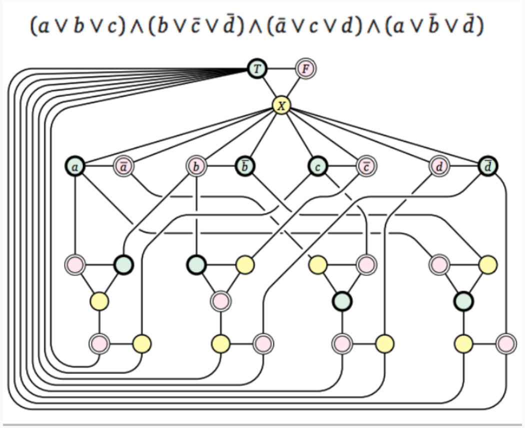
Correctness Proof

If our knapsack instance has a solution, then (S, T) has a subset sum.

- For each item i in the knapsack solution, add s_i to the subset sum solution S'
- We know that the total weight of all items in the knapsack solution is at most $C = T$
- The total value of all items in the knapsack solution is at least $V = T$.
- We have $v_i = w_i = s_i$. So the sum of the weights and values of the knapsack solution are the same, and must be exactly T. This is also $\sum_{s_i \in S'} s_i$. $\square$

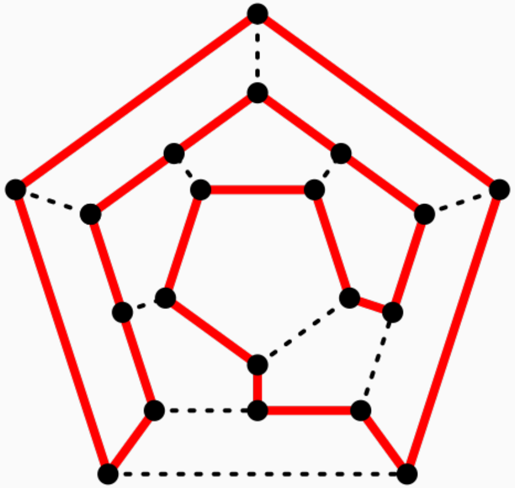
Other NP Complete Problems

Graph 3-coloring


$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$
T
F
X
a
$\bar{a}$
b
$\bar{b}$
c
$\bar{c}$
d
$\bar{d}$

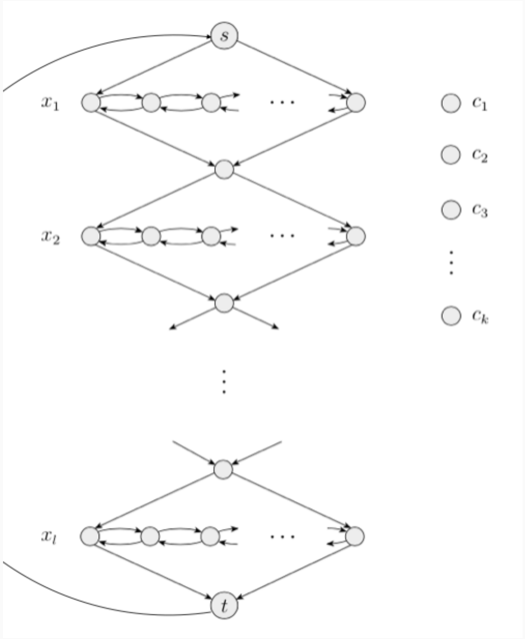
- Given a graph G
- Can we assign colors to each vertex of the graph (one of 3 colors) such that each edge has endpoints of different colors?
- Reduction idea: from 3SAT. Much like independent set, create “gadgets” which enforce 3SAT requirements.

Hamiltonian Cycle



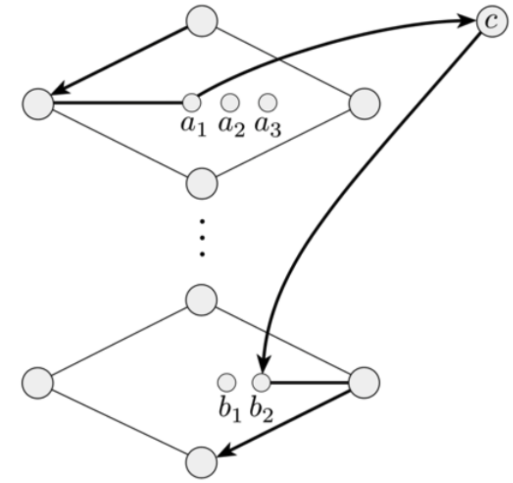
- Given a graph G
- Is there a simple cycle that visits every vertex exactly once?
- Also NP-complete: is there a simple path that visits every vertex exactly once?

Hamiltonian Cycle Reduction



- From 3SAT! Looks a little like independent set or 3-color
- Diagram on left: must choose if the path goes through the middle vertices left or right; corresponds to variable being positive or negative

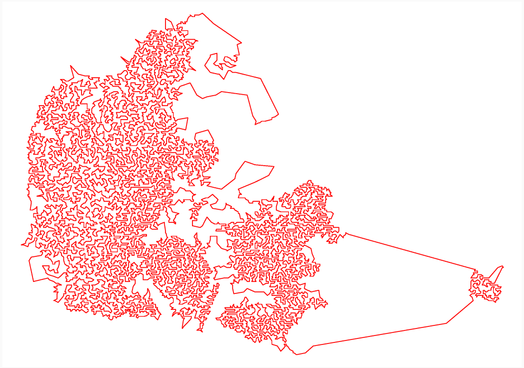
Hamiltonian Cycle Reduction



Such a cycle would never visit node a_2

- Need at least one "true" variable in each clause or you miss some vertices (see diagram)
- In Theory of Computation will see in a little more detail

Travelling Salesman Problem (TSP)

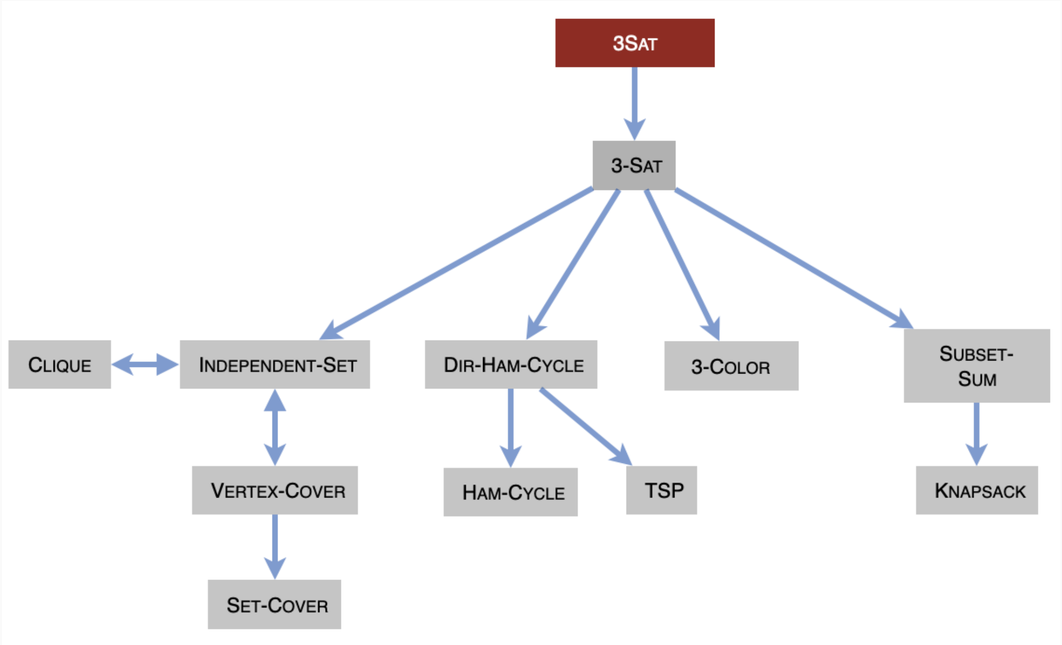


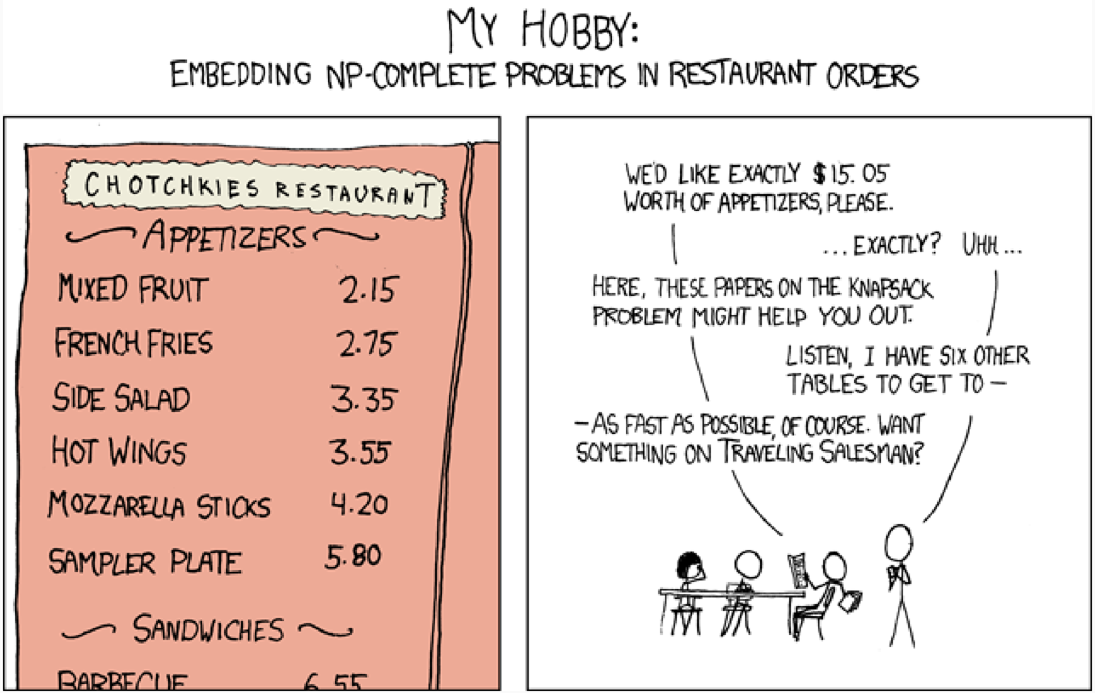
- What is the *shortest* path through a graph that visits every vertex?
- Decision version: is there a path that visits every vertex of length k?
- Classic NP-complete problem

TSP Reduction Summary

- From Hamiltonian Cycle. *Practice:* how can we use Hamiltonian cycle to prove TSP NP-hard?
- Given a Hamiltonian Cycle instance G
- Create a complete graph G'. Weight of an edge e is 1 if $e \in G$, 2 if $e \notin G$.
- Proof summary: G' has a TSP tour of length $k = n$ if and only if G has a Hamiltonian cycle

Problems We Have Proven NP-complete




MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS
CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
BARBECUE 6.55
WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO –
– AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?

Other/Fun NP-Hard Problems

NP-hard Problems in Other Areas

- Biology/Chemistry: Protein folding
- Civil Engineering: Urban traffic flow equilibrium
- Economics: Arbitrage in financial markets with friction
- Mechanical Engineering: Computing turbulence in sheared flows
- Physics: Partition function of 3D Ising Model
- Political Science: Computing the Dodgson winner of an election
- Statistics: Optimal experimental design

Fun NP-hard Games

- Minesweeper
- Candy Crush saga
- Rubik's Cube (2017 result; from Hamilton cycle)
- Super Mario Brothers (from 3-SAT; Aaron Williams has a paper on this)
- Tetris

Approximation Algorithms

Approximation Algorithms

- NP-hard problems are very important to solve
- Can’t get the *optimal* solution efficiently
- Idea: guarantee that we get a good solution—just not an optimal one

Simple Knapsack Variant

- Have a set of items with weight w_i, capacity C. (No value!)
- Goal: pick the subset of items with *maximum total weight*, subject to the total weight being $\leq C$
- Want our knapsack as "full as possible"
- Equivalent to classic knapsack with $v_i = w_i$ for each item; this is still NP-hard

Greedy Algorithm

- What is a good greedy algorithm for this problem?
- Repeatedly: pick the item with *largest* weight that does not overfill the knapsack.
- Let's do an example [On Board #1]. Items: $\{3, 4, 5, 6, 7, 10, 11, 12, 13, 15, 16, 19, 20\}$; $C = 47$
    - (Not obvious: can get 47 exactly using $\{3, 5, 19, 20\}$.)
- How long does this greedy algorithm take?

How bad can greedy be?

- It seems like it's usually pretty good
- Can we come up with an example where it's possible to get C, but greedy gets $C/2+1$?
- In particular: let's say $C=10$. Can we come up with an instance where greedy gets 6, but it's possible to get 10?
    - $\{6,5,5\}$
- In general (assume C is divisible by 2): $C/2+1, C/2, C/2$.

Greedy is an Approximation Algorithm

- If the best solution has weight OPT, greedy gets at least $OPT/2$.
- We say that greedy is a *2-approximation*. It's at most a factor 2 off of the optimal cost
- How can we prove this?

Greedy is a 2-Approximation Algorithm

- **Case 1:** First, let's say there is an item of weight $\geq C/2$
  - Greedy will achieve at least $C/2$
  - $C \geq OPT$, so greedy gets $\geq C/2 \geq OPT/2$
- **Case 2:** Now, let's say all items have weight $< C/2$
  - If greedy uses all items, then greedy achieves OPT
  - If greedy stops at total weight X, then there is no item left with weight $\leq C - X$. But then $X \geq C/2$.

Approximation Algorithms

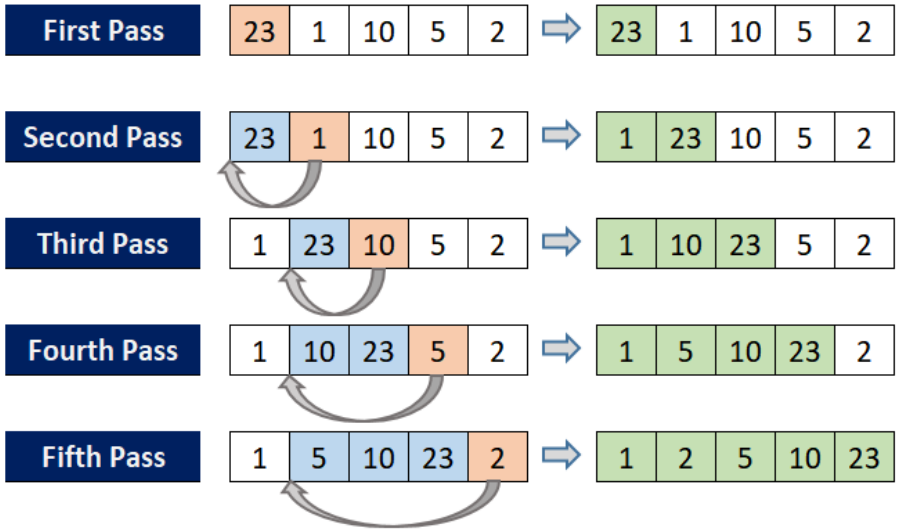
- When we can't get the best solution, can get a guarantee on *how close* we are
- We saw a simple, efficient Knapsack 2-approximation algorithm. Can we do better?
- Yes! Can get *any* $1 + \varepsilon$ approximation in polynomial time. (Even with both weights and values!)
    - Surprisingly simple algorithm
- What about other NP-complete problems? Can we approximate them?
- Sometimes...
    - Vertex cover: simple 2-approximation algorithm. (Probably) can't do better!
    - Clique: cannot approximate to $n^{1-\delta}$ for any $\delta > 0$ unless P = NP.

Algorithms lectures Completed!

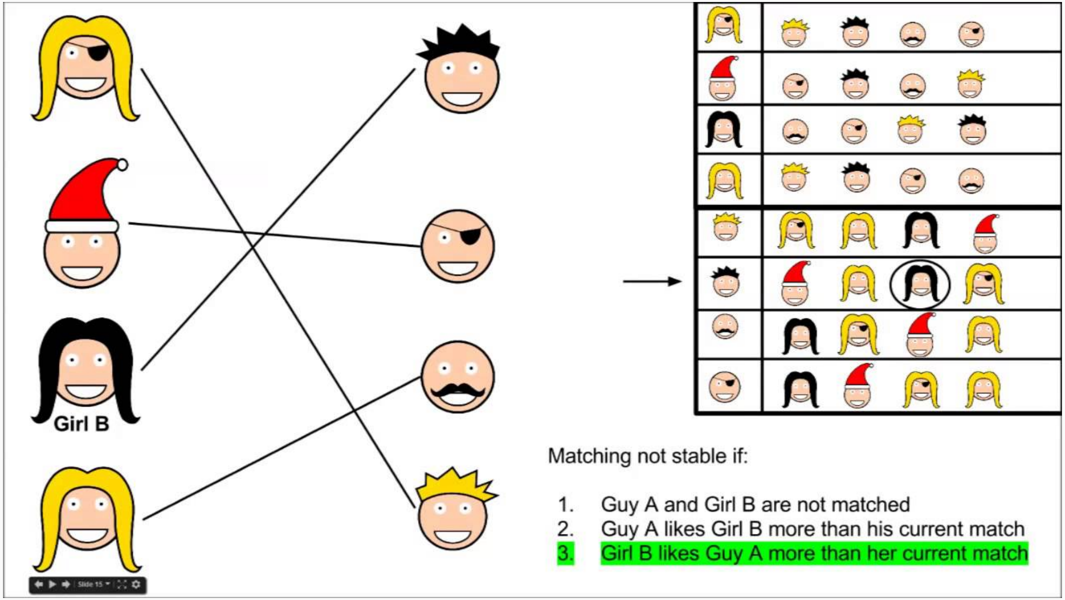
Looking Back at the Class

- You’ve learned a lot!

Proof of Correctness and Asymptotics



Stable Matchings

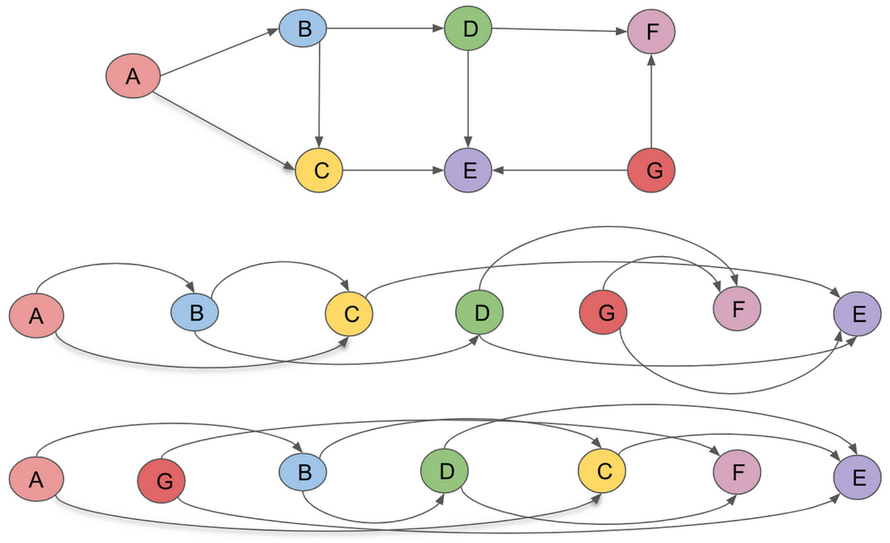


Matching not stable if:

1. Guy A and Girl B are not matched
2. Guy A likes Girl B more than his current match
3. Girl B likes Guy A more than her current match

Graph Traversal Algorithms

Ex: topological sort



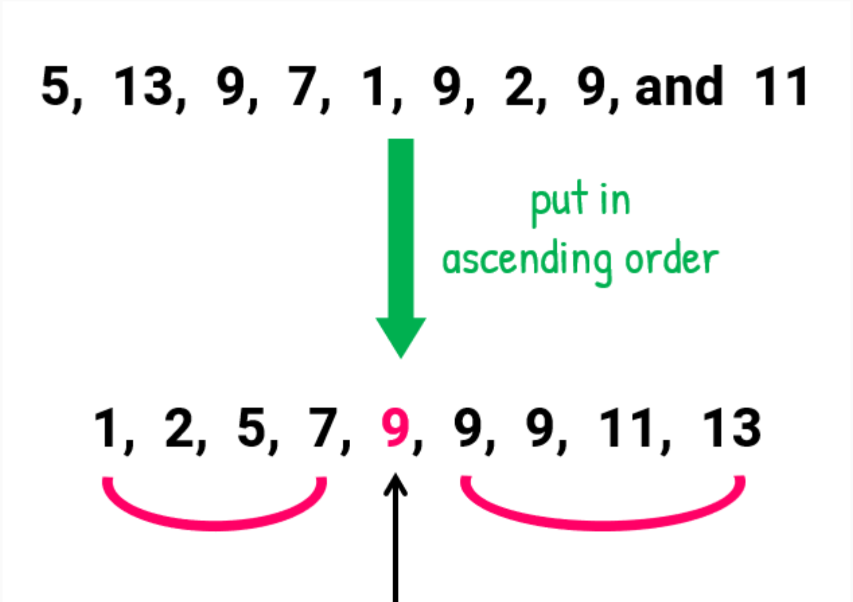
Topological Sort

Greedy Algorithms

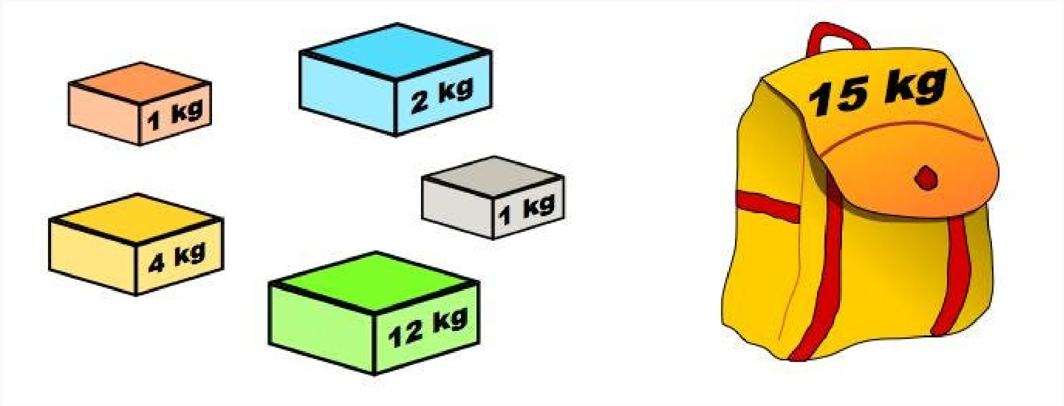
Ex: optimal car filling

Divide and Conquer

Ex: finding median without sorting. (Also Merge Sort, etc.)



Dynamic Programming, Network Flows



NP-Hardness

What can we *not* solve efficiently?

Review Thursday!

- Come with questions
- Assignment questions are always a good option; I’ll come with a few suggestions as well

SCS Forms

SCS Forms

- You know the drill (but let me know if you have questions)
- Fill out forms on Glow; course called “Course Evaluations”
- Blue sheets are for me only; rest of form is given to admin etc. All are anonymous; I can’t see anything until I submit grades
- If you can then do them now, but later is OK if necessary. They close 8am on May 15th.

SCS Forms Speech

Every term, Williams asks students to participate in end-of-semester course evaluations. Your feedback will help improve this course for other students taking it in the future, and help shape the Computer Science curriculum.

You may skip questions that you don't wish to answer, and there is no penalty for choosing not to participate. All of your answers are confidential and I will only receive a report on your responses after I have submitted all grades for this course. While evaluations are open, I will receive information on how many students have filled out the evaluations, but I won't be told which of you have and haven't completed them. I won't know which responses are associated with which student unless you identify yourself in the comments.

To access the online evaluations, log into Glow (`glow.williams.edu`) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "Course Evaluations." Click on this and then follow the instructions on the screen. If you have trouble finding the evaluation, you can ask a classmate or reach out to Institutional Research at `ir@williams.edu`. The evaluations are open to you from now through the end of reading period. If you haven't filled it out by the beginning of reading period, you will start receiving email reminders.