".

Greedy Algorithms

Sept 11, 2025

Greedy Alg =
- build the solution step by step
- at each step, choose the option that makes progress and (i) "makes the most progress" and/or (ii) "makes progress and costs the least"

E.g. Making change in coin denominations using the fewest coins

1. $(\overset{\text{amount}}{32}, \overbrace{[25, 10, 5, 1]}^{\text{coin denominations}})$

2. $(34, [30, 17, 1])$

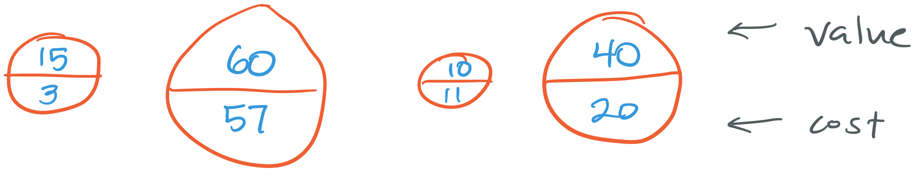
* Some interesting notes. Greedy works on some denomination sets and not on others. Characterizing the denom sets it works on is an open question.

Choosing to be greedy is also sometimes an art....

- What metric should the alg be greedy about?

efficient ≈ myopi

Knapsack problem:



Find the set that maximizes value while keeping cost ≤ N

- take the item that will not overrun cost and .... has min cost?
  has max value?
  has max value/cost ratio?

[These are myopic in that they are local conditions (not global)

- global - "which selection is part of an optimal solution?"
- local/myopic - "which value/cost ratio is smallest?"

* Do any of these work? Does Dynamic Pgrming work on Knapsack?

Given that you have selected a greedy Strategy you think will work, how do you prove that the result is always optimal?

Usually Either:

- Show that, at each step, it has built a partial solution that is as good as any other of same size or set of constraints

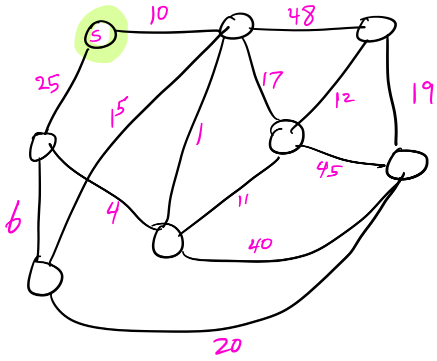
OR

- Devise an "Exchange argument"
  - take any solution
  - show a series of exchanges that can transform the given solution into a greedy on without diminishing the quality

Eg. from CSCI 260:

Dyjkstra's Shortest Paths Alg

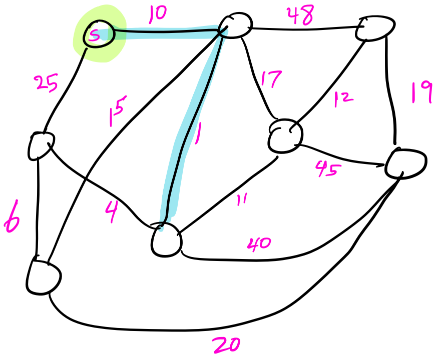
Prim or Kruskal's Min Spanning Tree Algs



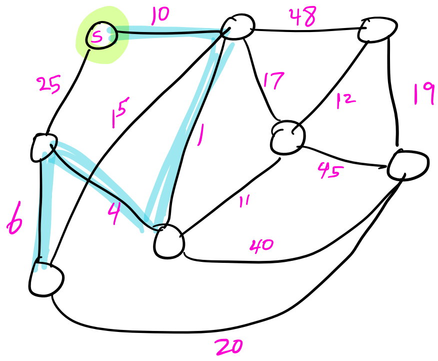
Eg. from CSCI 260 :

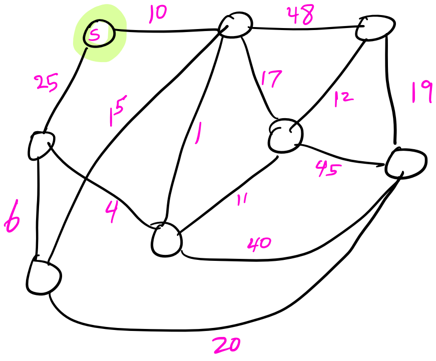
Dyjkstra's Shortest Paths Alg

Prim or Kruskal's Min Spanning Tree Algs



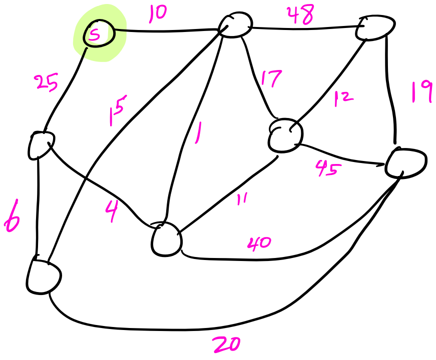
What is the loop invariant?



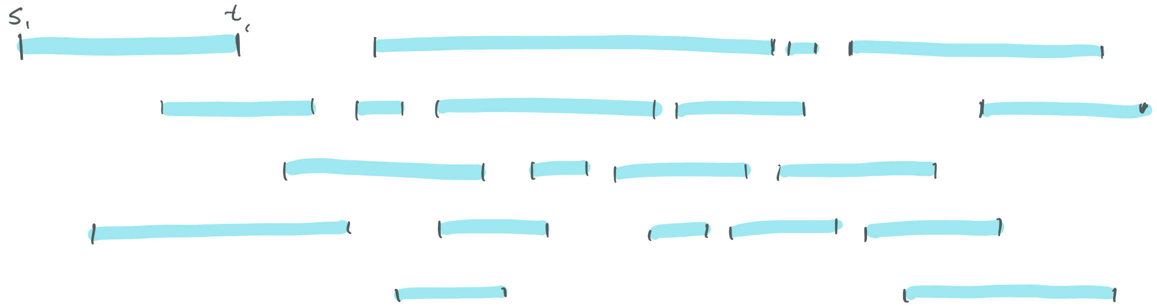


Loop Invar: "After adding K edges to T, T is a least-weight subtree of G induced on V(T)."

Proof is "step by step there is an opt solution that extends this solution."



Interval Scheduling



Given a set $\{(s_1,t_1),(s_2,t_2),\dots,(s_n,t_n)\}$ of intervals

$s_i \in \mathbb{Q}$, $s_i \le t_i$ $t \in \mathbb{Q}$

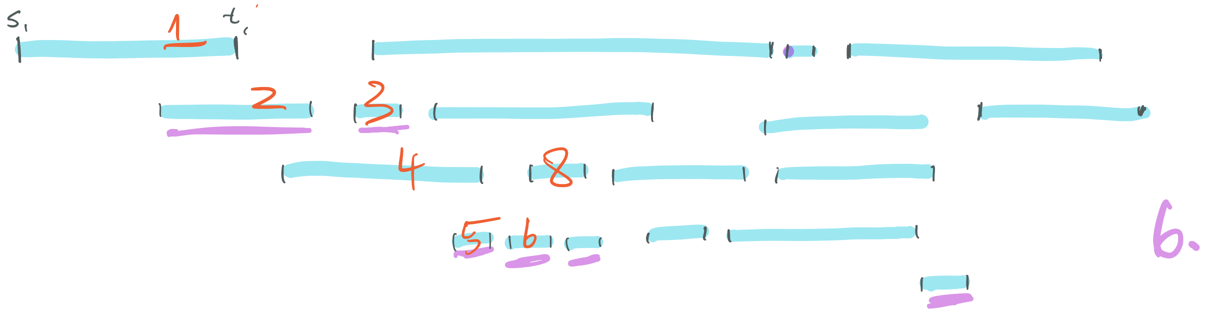
Def^n: Two intervals are compatible if they do not overlap.

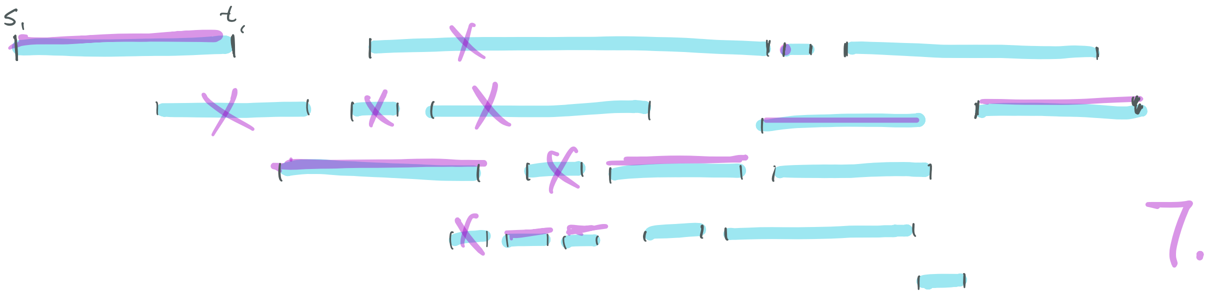
Def^n: A schedule is a set of intervals that are pairwise compatible.

Find the largest cardinality schedule.

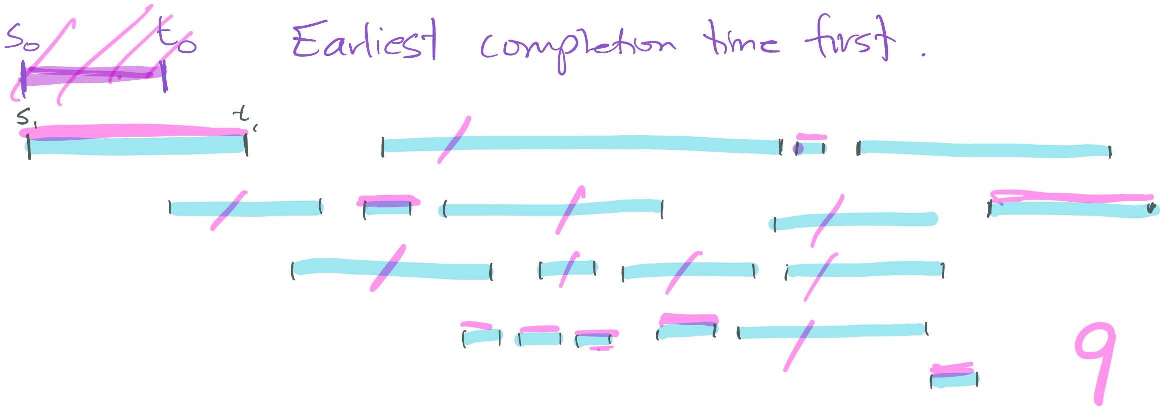
What greedy approaches might we consider?

- fewest incompatibilities first
- earliest start first ✗ counter
- shortest job first. ✗ counter
- largest interval - substitutions - first.





Earliest completion time first.



```
Max Interval Schedule (S a set of intervals)
        $A = \emptyset$
        while $S \neq \emptyset$
                pick interval $I$ in $S$ that has earliest completion time
                        and add $I$ to $A$
                remove from $S$ all intervals incompatible with $I$
```

```
    return $A$
```

To show this alg finds an optimal solution:

1. It is a schedule, because . . .

2. It is optimal, because . . .

Lets assume S is non-empty.

Let $I_1, I_2, \dots, I_k$ be the intervals in the order added to A

Let $J_1, J_2, \dots, J_{k'}$ be any other schedule, in the order of their start (= order of completions) $k' > 0$

For any interval X, let $s(X)$ be start, $t(X)$ be complete.

Claim: $\forall r \le k'$, I_r exists and $t(I_r) \le t(J_r)$

Proof: By induction on r.

$r = 1$: By selection at line 3, $t(I_1) \le t(J_1)$

Let $r > 1$, $r \le k'$

Ind Hyp: $t(I_{r-1}) \le t(J_{r-1})$

By IndHyp, I_{r-1} completes at least as early as J_{r-1}.

$\therefore$ J_r is compatible with I_{r-1}

$\therefore$ I_r exists, and $t(I_r) \le t(J_r)$, by selection criterion line 3

$\therefore$ By Induction on r, $t(I_r) \le t(J_r)$ $\forall r \le k'$ ▨

Corollary: Earliest Completion First is optimal.

∴ Greedily selecting intervals that are

a) compatible with set already selected

b) within that set, have earliest completion time

will lead to an optimal schedule (where optimal = max # of intervals)

Implementation details that lead to complexity claim:

- Sort by completion time $O(n \log n)$

$$\begin{array}{r} O(n \log n) \\ + \; O(n) \\ \hline O(n \lg n) \end{array}$$