Algorithm notes - CLRS

Greedy v/s Dynamic Programming

Greedy have many similarities to DP but one salient difference b/w greedy algorithms and DP is that in greedy, we use optimal substructure in a top-down fashion. Instead of first finding optimal solutions to subproblems and then making a choice, greedy first make a choice - the choice that looks best at the time - and then solve a resulting subproblem

Dynamic Programming Bottom-up approach

- → optimal substructure (warning: subtleties)
- → Overlapping Subproblems [Efficient]
- → Reconstructing an optimal solution
- → Memoization.

Greedy Top-down fashion

- → Greedy algorithm doesnt always yield optimal solutions

CLRS 16.1 An activity-selection Problem (Scheduling problem)

The activity selection problem is to select a maximum-size subset of mutually compatible activities

Suppose we have a set $S = \{a_1, a_2, \ldots a_n\}$ of n proposed activities that wish to use a resource, such as a lecture hall, which can be used by only one activity at a time. Each activity a_i has a start time s_i and a finish time f_i, where $0 \leq s_i < f_i < \infty$.

Elements of Greedy algorithm (CLRS 16.2)

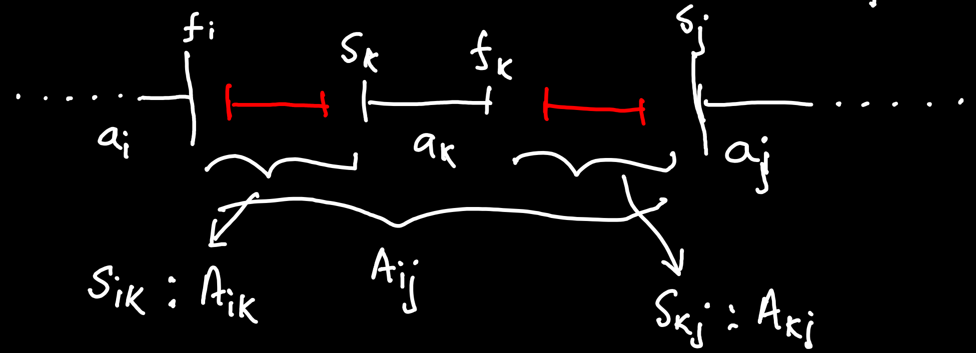
1. Determine the optimal substructure of the problem
2. Develop the recursive solution
3. Prove that at any stage of the recursion, one of the optimal choices is the greedy choice. Thus, it is always safe to make the greedy choice.
4. Show that all but one of the subproblems induced by having made the greedy choice is empty
5. Develop a recursive algorithm that implements the greedy strategy
6. Convert the recursive algorithm to an interative algorithm

→ Greedy cannot handle overlapping problem, subproblems should be independent of each other.

Activity Scheduling Problem

Dynamic Programming Analysis

$$S_{ij} = \{ a_k \in S : f_i \leq s_k < f_k \leq s_j \}$$



$$A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj} \Rightarrow |A_{ij}| = |A_{ik}| + 1 + |A_{kj}|$$

$$c[i,j] = \begin{cases} 0 & \text{if } S_{ij} = \phi \\ \max\limits_{a_k \in S_{ij}} \{ c[i,k] + c[k,j] + 1 \} & \text{if } S_{ij} \neq \phi \end{cases}$$

Greedy choice Analysis

$$S_k = \{ a_i \in S : s_i \geq f_k \}$$

choose $a_1 \rightarrow$ solve $S_1 \ldots\ldots$

choose $a_k \rightarrow$ solve $S_k \ldots\ldots$

Theorem : If $S_k \neq \phi$ and a_m has earliest finish time in S_k, then a_m is included in some optimal solution.

Let A_k be optimal solution to S_k

Let $a_j \in A_k$ have earliest finish time in A_k

If $j = m$, we are done

Let $A'_k = \underbrace{(A_k - \{a_j\}) \cup}\{a_m\}$

It is legal for this substitution by hypothesis

<u>Hypothesis</u>

a_m has earliest finish time in S_k

MIT 6.046 Lecture-12

contains all the vertices

Greedy algorithms → Minimum $\underbrace{\text{spanning tree}}$

subset of edges of G that form a tree & hit all vertices of G

MST: Given graph $G=(V,E)$ & edge weights $w: E \to \mathbb{R}$

Find a spanning tree T of minimum weight $= \sum_{e \in T} w(e)$

Greedy properties

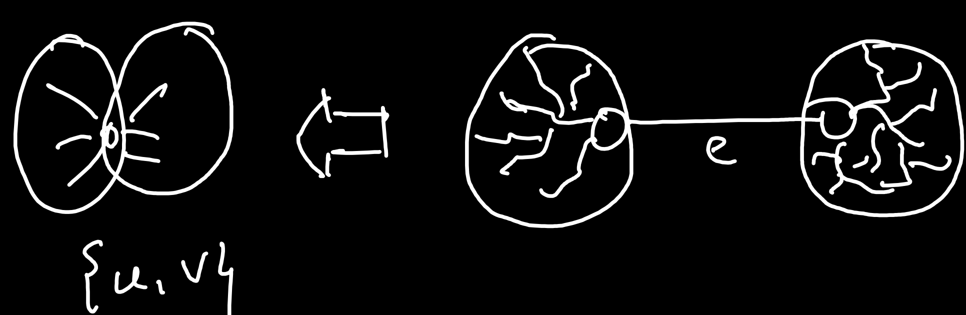
1. Optimal substructure : Optimal solution to problem incorporates optimal solution to subproblems
2. Greedy choice property : Locally optimal choices lead to globally optimal solutions

Optimal substructure for MST

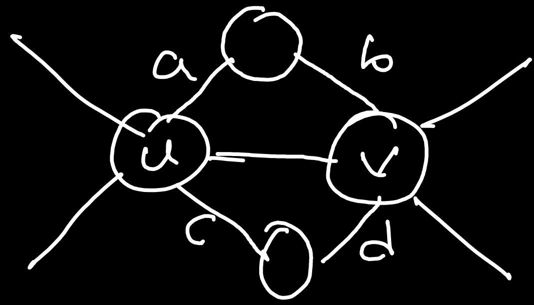
If $e = \{u, v\}$ is an edge of some MST

→ Contract e

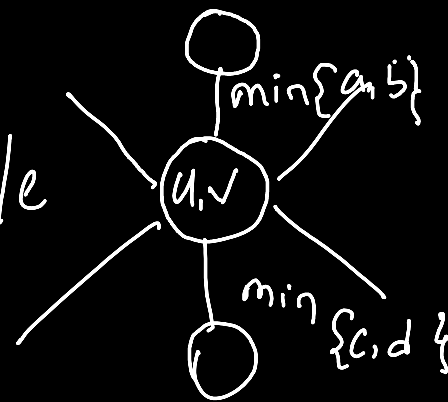
merge u & v



G:



$\Rightarrow$ G/e



→ if T' is a MST of G/e
then $T' \cup \{e\}$ is the MST of G

Dynamic Program

→ guess edge in a MST
→ contract e
→ recurse
→ de contract
→ add e to MST

Exponential time

Proof for optimal substructure

Say MST $T^* \ni e$

$\Rightarrow$ $T^* - e$ is spanning tree of G/e

$w(T') \leq w(T^* - e)$

$$w(T' \cup \{e\}) = w(T') + w(e)$$
$$\leq w(T^* - e) + w(e) = w(T^*)$$

The proposed $T' \cup \{e\}$ has less weight than T^*. Hence $T' \cup \{e\}$ is the MST

Greedy-choice property for MST

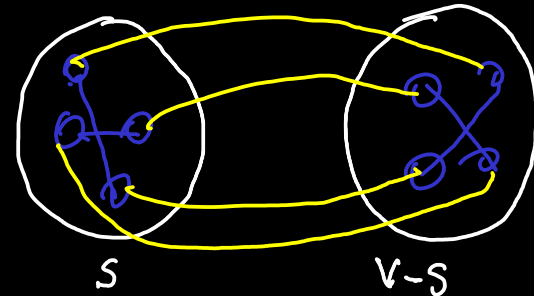
→ consider any cut $(S, V-S)$

→ say e is a least-weight edge crossing cut

$e = \{u, v\}, u \in S, v \in V-S$

→ then $e \in$ some MST

crossing edge



Proof: cut & paste argument [universal way of proving Greedy algorithm]

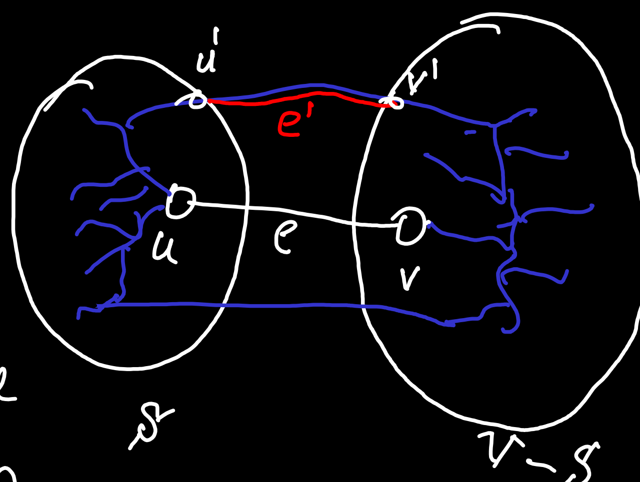
→ Let T^* be a MST of G

→ Say $e \notin T^*$

→ must be $e' \in T^*$ crossing the cut

→ $\underbrace{\{T^* - e'\} \cup \{e\}}_{}$ is Spanning tree → proved from graph theory part

proving ↓ is minimum is done by greedy choice property



$$w(T^* - e' \cup \{e\}) = w(T^*) - w(e') + w(e)$$
$$\leq w(T^*)$$

$w(e) \leq w(e')$ → MST

Hence we have proved Greedy choice property i.e $e \in$ MST

Only modified edges crossing cut $(S, V-S)$

Prim's algorithm [Dijstra's like algo]

→ Maintain priority queue Q on $V-S$ where

$v.key = \min \{ w(u,v) \mid u \in S \}$

→ initally Q stores V

→ $s.key = \phi$ for arbitrary start vertex $s \in V$

→ for $v \in V$, $v.key = \infty$

→ until Q empty :

→ u = Extract-min(Q) ($\Rightarrow$ add u to S)

→ for $v \in Adj[u]$

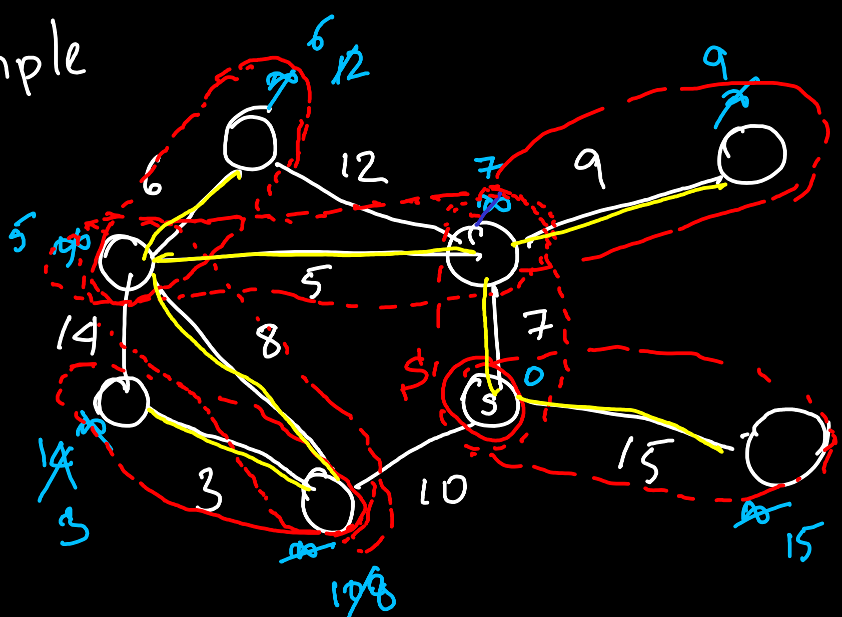
→ if $v \in Q$ & $w(u,v) < v.key$:

→ $v.key = w(u,v)$

→ $v.parent = u$

→ return $\{ \{ v, v.parent \} \mid v \in V \}$

Example



Proof of Correctness

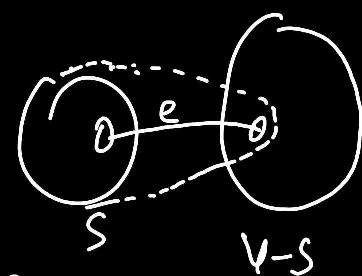
* Invariant → $v.key = \min\{w(u,v) \mid u \in S\}$ (Induction)

* Tree T_S within $S \subseteq$ (some) MST of G [Induction]

  → By induction : MST $T^* \supseteq T_S$

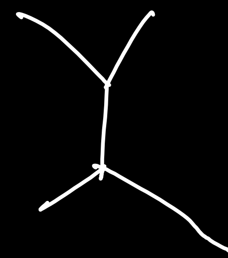
  → $T_S \to T_{S'} = T_S \cup \{e\}$

  → By greedy choice property modify T^* to include e & T_S



Time : Same as Dijkstra $O(V \lg V + E)$

Kruskal's Algorithm :



- maintain connected components in MST-so-far T in Union-Find structure
- $T = \emptyset$
- for $v \in V$ make_set(v)
- sort E, by weight [increasing order]
- for $e = \{u, v\} \in E$
  - if Find_set(u) $\neq$ Find-set(v):
    - $T = T \cup \{e\}$
    - Union(u, v)

Running time : $O(\text{sort}(E) + E\,\alpha(V) + V)$

Proof of correctness: invariant : Tree T so far $\subseteq$ some MST T^*

- assume $T \subseteq T^*$
- $T \to T' = T \cup \{e\}$
- use greedy-choice property with $S = CC(u)$

$\Rightarrow T' \subseteq T^{*\prime}$