Greedy Algorithms

CSC 349

Christopher Siu

The Traveling Salesperson Problem

NEARESTNEIGHBOR$(G = (V, E))$

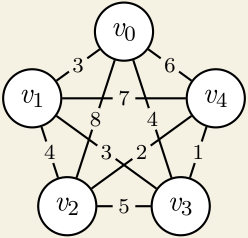
Input: A complete, weighted graph G
Output: The weight of a Hamiltonian cycle in G

```
for all v ∈ V do
    let v be "unexplored"
let s = v be "explored", any vertex in V, and W be 0
while there exist "unexplored" vertices in V do
    let e = (v, u) be the lightest edge such that u is "unexplored"
    let v be u and W be W + w_e
return W + w_e, where e = (v, s)
```

The Traveling Salesperson Problem

Example

Given:



…the Hamiltonian cycle of minimum weight is $(v_0, v_1, v_2, v_4, v_3, v_0)$, weight 14. $\textsc{NearestNeighbor}$ returns $(v_0, v_1, v_3, v_4, v_2, v_0)$, weight 17, and does *not* solve the Traveling Salesperson Problem.

Greedy Algorithms

Definition

A **greedy** algorithm is one that assumes a locally optimal choice will always lead to the globally optimal solution.

- A divide and conquer approach may not improve complexity.
- A greedy approach may be *incorrect*.

Example

ShortestPath (correctly) assumes that at least one of the lightest paths from a vertex will always traverse the lightest available path.

Example

NearestNeighbor (incorrectly) assumes that the lightest Hamiltonian cycle will always traverse the lightest incident edge.

Minimum Spanning Trees

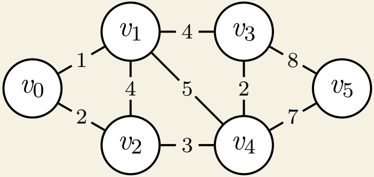
Minimum Spanning Trees

Consider the following problem:

- Given a connected, weighted graph, find an "MST", a spanning tree of minimum weight.

Example

Given:



...the minimum spanning tree has weight 15.

Minimum Spanning Trees

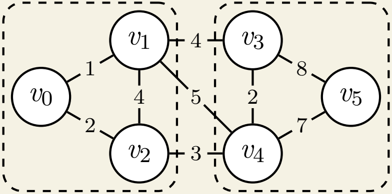
Definition

A **cut** is a partitioning of every vertex into two disjoint sets.

- Given a cut, its **cut set** is the set of all edges connecting vertices in different sets.

Example

Given:



...the cut set is $\{(v_1, v_3), (v_1, v_4), (v_2, v_4)\}$.

Minimum Spanning Trees

The Cut Property

If e is the edge of strictly minimum weight in a cut set, then e must be in every MST.

- If lighter edges are considered first, then an edge that would connect previously unreachable vertices is always optimal.

The Cycle Property

If e is the edge of strictly maximum weight in a cycle, then e cannot be in any MST.

- If heavier edges are considered last, then an edge that would connect already reachable vertices is never optimal.

Kruskal's Algorithm

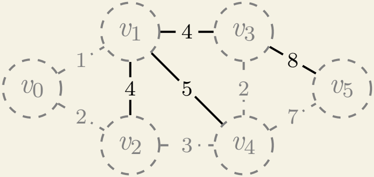
$\textsc{MinimumSpanningTree}(G = (V, E))$

Input: A connected, weighted graph G
Output: A minimum spanning tree of G

```
let T be an empty tree and S be an empty disjoint set
for all v ∈ V do
    Make a new subset containing v in S
let A be E, sorted in increasing order by weight
for all e = (u, v) ∈ A do
    if u and v are in different subsets in S then
        Union the subsets containing u and v in S
        Add e to T
return T
```

Kruskal's Algorithm

Example



1. Add $e = (v_0, v_1),\ w_e = 1.$
2. Add $e = (v_0, v_2),\ w_e = 2.$
3. Add $e = (v_3, v_4),\ w_e = 2.$
4. Add $e = (v_2, v_4),\ w_e = 3.$
5. Skip $e = (v_1, v_2),\ w_e = 4.$
6. Skip $e = (v_1, v_3),\ w_e = 4.$
7. Skip $e = (v_1, v_4),\ w_e = 5.$
8. Add $e = (v_4, v_5),\ w_e = 7.$
9. Skip $e = (v_3, v_5),\ w_e = 8.$

Kruskal's Algorithm

Example

- Sorting E has complexity $O(|E| \log |E|)$.
- Each vertex appears in the disjoint set exactly once.
- Each edge causes a disjoint set operation at most thrice.

Assuming a tree-based implementation of the disjoint set, MinimumSpanningTree has complexity $O(|E| \log |E|)$.

- Disjoint sets can be implemented as trees: two nodes have the same root if and only if their elements are in the same subset.
- With a hash table implementation of the disjoint set, Kruskal's algorithm has complexity $O(|V| |E|)$.

Exchange Arguments

Theorem

Suppose $\textsc{MinimumSpanningTree}$ returns T and there exists an optimal MST $\textsc{Opt}$.

Lemma

If T differs from $\textsc{Opt}$ by i edges, then $\textsc{Opt}$ can be transformed into T without increasing its weight.

Then T and $\textsc{Opt}$ have the same weight.

- Note that T need not include the same edges as $\textsc{Opt}$; it need only have the same sum total weight.
- Since the edges of $\textsc{Opt}$ can be "exchanged" to equal those of T without changing their weight, T has the same weight.

Activity Selection

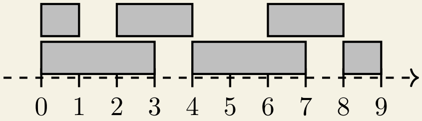
Activity Selection

Consider the following problem:

- An **activity** is a pair $a_i = (s_i, f_i)$ of a start time s_i and a finish time f_i, taking place during the time $[s_i, f_i)$.
- Given a set of activities, find a maximum subset of non-conflicting activities.

Example

Given $\{(2,4), (0,3), (0,1), (4,7), (6,8), (8,9)\}$:



...one of the maximum subsets is $\{(0,1), (2,4), (4,7), (8,9)\}$.

Activity Selection

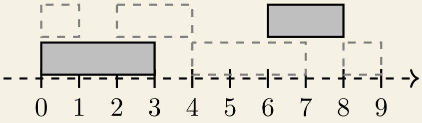
$\textsc{ActivitySelection}(S = \{a_0, a_1, \ldots, a_{n-1}\})$

Input: A finite, non-empty set S of n activities, where $a_i = (s_i, f_i)$
Output: A maximum subset of non-conflicting activities

let A be S, sorted in increasing order by finish time
let $a_j = (s_j, f_j)$ be the first activity in A and X be $\{a_j\}$
for all $a_i = (s_i, f_i) \in A$ **do**
    if $s_i \geq f_j$ **then**
        let X be $X \cup \{a_i\}$
        let a_j be a_i
return X

Activity Selection

Example



1. Select $(0, 1)$.
2. Skip $(0, 3)$.
3. Select $(2, 4)$.
4. Select $(4, 7)$.
5. Skip $(6, 8)$.
6. Select $(8, 9)$.

Example

- Sorting S has complexity $O(n \log n)$.
- Collectively considering activities has complexity $O(n)$.

ActivitySelection has complexity $O(n \log n)$.

Greedy Stays Ahead

Theorem

Suppose $\textsc{ActivitySelection}$ returns $X = (x_0, x_1, \ldots, x_{n-1})$ and there exists an optimal selection $\textsc{Opt} = (y_0, y_1, \ldots y_{m-1})$, both sorted in increasing order of finish time.

Lemma

For all $i < n$, $f_{xi} \leq f_{yi}$.

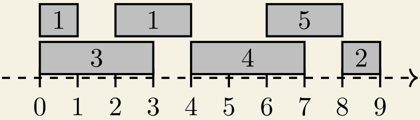
Then $n = m$.

- Note that X need not include the same activities as $\textsc{Opt}$; it need only have as many activities as $\textsc{Opt}$.
- By "staying ahead" of $\textsc{Opt}$, X always has the option of including the same activities.

Weighted Activity Selection

Example

Given:



...$\textsc{ActivitySelection}$ returns $\{(0,1), (2,4), (4,7), (8,9)\}$, which is a subset of maximum cardinality but not maximum weight.