CSE 421

Longest Increasing Subsequence, Shortest Paths with Neg Weights, Network Flow

Shayan Oveis Gharan
Longest Path in a DAG
Longest Path in a DAG

**Goal:** Given a DAG G, find the longest path.

**Recall:** A directed graph G is a DAG if it has no cycle.

This problem is NP-hard for general directed graphs:
- It has the Hamiltonian Path as a special case.
Suppose we have labelled the vertices such that \((i, j)\) is a directed edge only if \(i < j\).

Let \(OPT(j)\) = length of the longest path ending at \(j\)

Suppose \(OPT(j)\) is \((i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k), (i_k, j)\), then

Obs 1: \(i_1 \leq i_2 \leq \ldots \leq i_k \leq j\).

Obs 2: \((i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k)\) is the longest path ending at \(i_k\).

\[
OPT(j) = 1 + OPT(i_k).
\]
Suppose we have labelled the vertices such that \((i, j)\) is a directed edge only if \(i < j\).

Let \(OPT(j) = \text{length of the longest path ending at } j\)

\[
OPT(j) = \begin{cases} 
0 & \text{If } j \text{ is a source} \\
1 + \max_{i: (i,j) \text{ an edge}} OPT(i) & \text{o.w.}
\end{cases}
\]
Let $G$ be a DAG given with a topological sorting: For all edges $(i,j)$ we have $i<j$.

Initialize $\text{Parent}[j]=-1$ for all $j$.

Compute-$\text{OPT}(j)${
\begin{align*}
\text{if} & \ (\text{in-degree}(j)==0) \\
& \text{return} \ 0 \\
\text{if} & \ (M[j]==\text{empty}) \\
& \text{M}[j]=0; \\
& \text{for all edges} \ (i,j) \\
& \quad \text{if} \ (M[j] < 1+\text{Compute-}\text{OPT}(i)) \\
& \quad \quad \text{M}[j]=1+\text{Compute-}\text{OPT}(i) \\
& \quad \text{Parent}[j]=i \\
\text{return} & \ M[j]
\end{align*}
}

Let $M[k]$ be the maximum of $M[1], \ldots, M[n]$

While $\text{(Parent}[k]!=-1)$
\begin{align*}
& \text{Print k} \\
& \text{k=Parent}[k]
\end{align*}
Longest Increasing Subsequence
Longest Increasing Subsequence

Given a sequence of numbers
Find the longest increasing subsequence

41, 22, 9, 15, 23, 39, 21, 56, 24, 34, 59, 23, 60, 39, 87, 23, 90

41, 22, 9, 15, 23, 39, 21, 56, 24, 34, 59, 23, 60, 39, 87, 23, 90
DP for LIS

Let $OPT(j)$ be the longest increasing subsequence ending at $j$.

**Observation:** Suppose the $OPT(j)$ is the sequence $x_{i_1}, x_{i_2}, \ldots, x_{i_k}, x_j$

Then, $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ is the longest increasing subsequence ending at $x_{i_k}$, i.e., $OPT(j) = 1 + OPT(i_k)$

$$OPT(j) = \begin{cases} 1 & \text{if } x_j > x_i \text{ for all } i < j \\ 1 + \max_{i:x_i < x_j} OPT(i) & \text{o.w.} \end{cases}$$

**Remark:** This is a special case of Longest path in a DAG: Construct a graph $1, \ldots, n$ where $(i, j)$ is an edge if $i < j$ and $x_i < x_j$. 
Shortest Paths with Negative Edge Weights
Shortest Paths with Neg Edge Weights

Given a weighted directed graph $G = (V, E)$ and a source vertex $s$, where the weight of edge $(u,v)$ is $c_{u,v}$

Goal: Find the shortest path from $s$ to all vertices of $G$.

Recall that Dijkstra’s Algorithm fails when weights are negative
Observation: No solution exists if $G$ has a negative cycle.

This is because we can minimize the length by going over the cycle again and again.

So, suppose $G$ does not have a negative cycle.
DP for Shortest Path

**Def:** Let \( OPT(v, i) \) be the length of the shortest \( s - v \) path with at most \( i \) edges.

Let us characterize \( OPT(v, i) \).

**Case 1:** \( OPT(v, i) \) path has less than \( i \) edges.

- Then, \( OPT(v, i) = OPT(v, i - 1) \).

**Case 2:** \( OPT(v, i) \) path has exactly \( i \) edges.

- Let \( s, v_1, v_2, \ldots, v_{i-1}, v \) be the \( OPT(v, i) \) path with \( i \) edges.
- Then, \( s, v_1, \ldots, v_{i-1} \) must be the shortest \( s - v_{i-1} \) path with at most \( i - 1 \) edges. So,
  \[
  OPT(v, i) = OPT(v_{i-1}, i - 1) + c_{v_{i-1}, v}
  \]
DP for Shortest Path

Def: Let $OPT(v, i)$ be the length of the shortest $s - v$ path with at most $i$ edges.

$$OPT(v, i) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{if } v \neq s, i = 0 \\ \min(OPT(v, i - 1), \min_{u: (u, v) \text{ an edge}} OPT(u, i - 1) + c_{u,v}) & \text{otherwise} \end{cases}$$

So, for every $v$, $OPT(v, ?)$ is the shortest path from $s$ to $v$.
But how long do we have to run?
Since $G$ has no negative cycle, it has at most $n - 1$ edges. So, $OPT(v, n - 1)$ is the answer.
Bellman Ford Algorithm

\[
\begin{align*}
\text{for } v=1 \text{ to } n \\
\text{ if } v \neq s \text{ then} \\
\quad M[v,0]=\infty \\
M[s,0]=0.
\end{align*}
\]

\[
\begin{align*}
\text{for } i=1 \text{ to } n-1 \\
\quad \text{for } v=1 \text{ to } n \\
\quad \quad M[v,i]=M[v,i-1] \\
\quad \quad \text{for every edge } (u,v) \\
\quad \quad \quad M[v,i]=\min(M[v,i], M[u,i-1]+c_{u,v})
\end{align*}
\]

**Running Time:** $O(nm)$

Can we test if $G$ has negative cycles? Yes, run for $i=1 \ldots 3n$ and see if the $M[v,n-1]$ is different from $M[v,3n]$
DP Techniques Summary

**Recipe:**
- Follow the natural induction proof.
- Find out additional assumptions/variables/subproblems that you need to do the induction.
- Strengthen the hypothesis and define w.r.t. new subproblems.

**Dynamic programming techniques.**
- Whenever a problem is a special case of an NP-hard problem an ordering is important:
  - Adding a new variable: knapsack.
  - Dynamic programming over intervals: RNA secondary structure.

**Top-down vs. bottom-up:**
- Different people have different intuitions.
- Bottom-up is useful to optimize the memory.
Network Flows
Soviet Rail Network

Network Flow Applications

Max flow and min cut.
• Two very rich algorithmic problems.
• Cornerstone problems in combinatorial optimization.
• Beautiful mathematical duality.

Nontrivial applications / reductions.
• Data mining.
• Open-pit mining.
• Project selection.
• Airline scheduling.
• Bipartite matching.
• Baseball elimination.
• Image segmentation.
• Network connectivity.
Minimum s-t Cut Problem

Given a directed graph \( G = (V, E) \) = directed graph and two distinguished nodes: \( s = \text{source}, t = \text{sink} \).

Suppose each directed edge \( e \) has a nonnegative capacity \( c(e) \).

Goal: Find a cut separating \( s, t \) that cuts the minimum capacity of edges.
s-t cuts

Def. An s-t cut is a partition \((A, B)\) of \(V\) with \(s \in A\) and \(t \in B\).

Def. The capacity of a cut \((A, B)\): \(\text{cap}(A, B) = \sum_{(u,v): u \in A, v \in B} c(u, v)\)

Capacity = 10 + 5 + 15 = 30
**s-t cuts**

**Def.** An **s-t cut** is a partition \((A, B)\) of \(V\) with \(s \in A\) and \(t \in B\).

**Def.** The **capacity** of a cut \((A, B)\): 
\[
\text{cap}(A, B) = \sum_{(u,v): u \in A, v \in B} c(u, v)
\]

**Diagram:**

- **Vertices:** \(s, 2, 3, 4, 5, 6, 7, t\)
- **Edges:**
  - \(s\) to \(3\): 5
  - \(3\) to \(4\): 15
  - \(4\) to \(5\): 9
  - \(3\) to \(2\): 10
  - \(2\) to \(5\): 15
  - \(5\) to \(t\): 10
  - \(3\) to \(6\): 8
  - \(6\) to \(7\): 30
  - \(6\) to \(t\): 15
  - \(7\) to \(t\): 10

**Capacity:** 
\[
\text{Capacity} = 9 + 15 + 8 + 30 = 62
\]
Minimum s-t Cut Problem

**Given** a directed graph $G = (V, E) = \text{directed graph}$ and two distinguished nodes: $s = \text{source, } t = \text{sink}$. 

Suppose each directed edge $e$ has a nonnegative capacity $c(e)$.

**Goal:** Find a s-t cut of minimum capacity

![Diagram of a directed graph](image)

Capacity $= 10 + 8 + 10 = 28$
**s-t Flows**

**Def.** An **s-t flow** is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$  
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

**Def.** The **value** of a flow $f$ is: $\nu(f) = \sum_{e \text{ out of } s} f(e)$

![Graph with annotated capacities and flows](image-url)
s-t Flows

Def. An s-t flow is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Def. The value of a flow $f$ is: $\nu(f) = \sum_{e \text{ out of } s} f(e)$

\[
\begin{array}{c}
2 \\
3 \\
4 \\
7 \\
5 \\
6 \\
+ \\
\end{array}
\]

\[
\begin{array}{c}
\text{capacity} \rightarrow 15 \\
\text{flow} \rightarrow 11 \\
10 \\
10 \\
10 \\
10 \\
10 \\
10 \\
\end{array}
\]

\[
\begin{array}{c}
3 \\
5 \\
4 \\
15 \\
8 \\
8 \\
4 \\
15 \\
\end{array}
\]

\[
\begin{array}{c}
10 \\
15 \\
6 \\
6 \\
15 \\
15 \\
15 \\
15 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
+ \\
\end{array}
\]

\[
\begin{array}{c}
10 \\
10 \\
10 \\
6 \\
6 \\
11 \\
11 \\
\end{array}
\]

Value = 24
Maximum s-t Flow Problem

Goal: Find a s-t flow of largest value.

Value = 28
Flow value lemma. Let $f$ be any flow, and let $(A, B)$ be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving $s$.

$$
\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)
$$

Value $= 24$