CSE 421

Longest Path in a DAG, LIS, Shortest Path with Negative Weights

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
DP for Longest Path in a DAG

Q: What is the right ordering?
Remember, we have to use that G is a DAG, ideally in defining the ordering

We saw that every DAG has a topological sorting
So, let’s use that as an ordering.
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 in the longest path ending at \(j\), last edge is \((i, j)\). Then, none of the \(i + 1, \ldots, j - 1\) are in this path since topological ordering. So,

\[OPT(j) = OPT(i) + 1.\]
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 \(j\)

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

\[
\text{Compute-OPT}(j)\{
    \text{if} \ (\text{in-degree}(j)==0)
    \quad \text{return} \ 0
    \text{if} \ (M[j]==\text{empty})
    \quad M[j]=0;
    \text{for} \ \text{all edges} \ (i,j)
    \quad M[j] = \max(M[j], 1+\text{Compute-OPT}(i))
    \quad \text{return} \ M[j]
\}
\]
Output \( \max(M[1], \ldots, M[n]) \)

**Running Time:** \( O(n + m) \)

**Memory:** \( O(n) \)

Can we output the longest path?
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-$\text{OPT}(j)$

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

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

While $(\text{Parent}[k] != -1)$

Print $k$

$k = \text{Parent}[k]$
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 \(OPT(j) = \text{length of the longest path ending at } j\)

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

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

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 \(OPT(j)\) = 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 Parent[j]=-1 for all j.

Compute-OPT(j) {
    if (in-degree(j)==0)
        return 0
    if (M[j]==empty)
        M[j]=0;
    for all edges (i,j)
        if (M[j] < 1+Compute-OPT(i))
            M[j]=1+Compute-OPT(i)
            Parent[j]=i
    return M[j]
}

Let M[k] be the maximum of M[1],...,M[n]

While (Parent[k]!=-1)
    Print k
    k=Parent[k]
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

OPT(\(i\)) = \(\max_{j : j < i, a_j < a_i} \\OPT(j) + 1\).

\[ \text{OPT}(i) \text{ length of longest inc subseq ending at } i. \]
DP for LIS

Let \( \text{OPT}(j) \) be the longest increasing subsequence ending at \( j \).

**Observation:** Suppose the \( \text{OPT}(j) \) is the sequence
\[
x_{i_1}, x_{i_2}, ..., x_{i_k}, x_j
\]

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

\[
\text{OPT}(j) = \begin{cases} 
  1 & \text{if } x_j > x_i \text{ for all } i < j \\
  1 + \max_{i: x_i < x_j} \text{OPT}(i) & \text{otherwise}
\end{cases}
\]

**Remark:** This is a special case of Longest path in a DAG: Construct a graph \( 1, ..., n \) 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 $s, v_1, v_2, ..., v_{i-1}, v$ be the $OPT(v, i)$ path with $i$ edges.
• Then, $s, v_1, ..., v_{i-1}$ 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.

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

So, for every $v$, $OPT(v, \cdot)$ 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

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

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

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

Can we test if \(G\) has negative cycles?
Yes, run for \(i=1\ldots3n\) and see if the \(M[v,n-1]\) is different from \(M[v,3n]\)
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