Outline

• Chapter 1:
  - 1.1, 1.2, 1.3, 1.4, 1.7, 1.8

• Chapter 3: Dynamic Programming

• Chapter 5: Binary Tree

• Chapter 6: Selected sections

• Chapter 2
Interval Scheduling Problem

- Given $n$ intervals of $v_i$ and lengths $w_i$. Find the subset of compatible intervals of maximum total value.
Interval Scheduling
Problem: Example

- \((s_1, f_1) = (0, 2)\)
- \((s_2, f_2) = (1, 3)\)
- \((s_3, f_3) = (0, 4)\)
- \((s_4, f_4) = (4, 7)\)
- \((s_5, f_5) = (6, 7)\)

Starting and finishing points of the interval
Interval Scheduling
Problem: Ideas

• Sort intervals by $f_i$
• Define $F(n)$ to be the maximum return
• $F(n)$ either
  - Includes $(s_n, f_n)$ or
  - Does not include $(s_n, f_n)$
• $p(n)$ subset $\sim F(n)$
  - $p(1) = 0$
  - $p(2) = 0$
  - $p(5) = 3$
Floyd-Warshall Algorithm

- Finds shortest paths between all pairs of nodes
- $d_{i,j} \geq 0$, but no negative cycles
- $d_{i,k} \geq d_{i,j} + d_{j,k}$

$$d_{i,k} \leftarrow \min(d_{i,k}, d_{i,j} + d_{j,k})$$

For $j = 1, 2, \ldots, n$ and all $i, k \neq j$

After $j = n$, we have the shortest distance between any pair of nodes.
Numerical Example

![Graph and Matrix](image)
Another Example

When $j = 1$...

If $d_{i,k} \leq d_{i,1} + d_{1,k}$ then $d_{i,k}$ remains unchanged.
If $d_{i,k} > d_{i,1} + d_{1,k}$ then $d_{i,k}$ is updated by the sum.

Then try $j = 2$, update with smaller values.
Basic Arcs

If $7 \sim 11$ is shortest
then $6 \sim 22$ is shortest
then $6 \sim 3$ is shortest

Basic Arc $(i, j)$: An arc $(i, j)$ is a basic arc iff the shortest path from $i$ to $j$ is the arc $(i, j)$. 
A shortest path must consist of basic arcs?
A path consisting of basic arcs must be shortest?
Counterexample showing that a path consisting of basic arcs is not necessarily shortest.
Proof that a shortest path must consist of basic arcs.

If we focus on an arbitrary shortest path, and we can get the shortest distance using basic arcs, then it is correct for all
In case of tie, which path will be picked by the computer?
We want the path also

We got the shortest distance, but not the shortest path

\[ p_{i,k} = k \] for all \( i, k \) to start

We have an associated array:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
\[ p_{i,k} = \begin{cases} 
  j & \text{if } d_{i,k} > d_{i,j} + d_{j,k} \\
  \text{same} & \text{if } d_{i,k} \leq d_{i,j} + d_{j,k} 
\end{cases} \]

This gives us some node \( j \) between \( i \) and \( k \), but we want the first node between \( i \) and \( k \).
Change the algorithm to:

\[ p_{i,k} = \begin{cases} 
  p_{i,j} & \text{if } d_{i,k} > d_{i,j} + d_{j,k} \\
  \text{same} & \text{if } d_{i,k} \leq d_{i,j} + d_{j,k}
\end{cases} \]
\[ j = 1 : \ p_{4,2} \leftarrow p_{4,1} \leftarrow 1 \]
\[ j = 2 : \ p_{1,3} \leftarrow p_{4,2} \leftarrow p_{4,1} \leftarrow 1 \]
\[ j = 3 : \ p_{4,3} \leftarrow p_{4,2} \leftarrow 1 \]

\[ 4 \sim 3 = p_{4,3} = 1 \]
\[ 1 \sim 3 = p_{1,3} = 2 \]
\[ 2 \sim 3 = p_{2,3} = 3 \]
### $d_{i,k}$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>0</td>
<td>2</td>
<td>$\infty$</td>
</tr>
<tr>
<td>3</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>$\infty$</td>
<td>0</td>
</tr>
</tbody>
</table>

### $j = 1$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$p_{4,2} \leftarrow p_{4,1} = 1$
Implementation

Given $n$ by $n$ array

$j = 1$:

Update $(n - 1)$ by $(n - 1)$ unshaded region.
$j = 2$: 

Update unshaded region.
Negative cycles?

How to find negative cycle?

Set $d_{i,i} = \infty$ to start, then run Floyd Warshall
Floyd -- Warshall

\[ d_{ik} := \min (d_{ik}, d_{ij} + d_{jk}) \]

for \( j = 1, ..., n, \) & all \( i,k \neq j \)

\[ d_{ik} := \max [d_{ik}, \min(d_{ij}, d_{jk})] \]

What other problems could be solved by modifying this triple operation?
Multiterminal Shortest Paths

\[ d_{ik} \leftarrow \min(d_{ik}, d_{ij} + d_{jk}) \]
for \( j = 1, 2, \ldots, n; \text{forall } i, k \neq j \)

\[ p_{ik} = \begin{cases} 
  p_{ij} & \text{if } d_{ik} > d_{ij} + d_{jk} \\
  \text{same} & \text{if } d_{ik} \leq d_{ij} + d_{jk} 
\end{cases} \]
Multiterminal Shortest Paths

\[ j=1 \]

\[ p_{42} = p_{41} \]

\[ j=2 \]
Multiterminal Shortest Paths

\( j = 3 \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Multiterminal Shortest Paths

j=4

Graph with nodes 1, 2, 3, 4 and edges labeled with numbers.
Weighted Knapsack Problem

• There are \( n \) types of items

• Every item has value \( v_i \)

• Every item has weight \( w_i \)

• The knapsack has capacity \( b \)

• Make input simple
Weighted Knapsack Problem

• Define Max Return Function $F_n(b)$

• $F_n(b) =$ Optimal value when capacity is $b$, and when $n$ types available

• Base Cases:
  - $b = 0$, $F_n(0) = 0$
  - $n = 1$, $F_1(b) = v_1 \left\lfloor b/w_1 \right\rfloor$
Weighted Knapsack Problem

- $F_k(y) = \text{Max Return when}$
  - $y = 0, 1, 2, \ldots, b$
  - $k = 1, 2, \ldots, n$

- $F_2(y) = \text{Max}[F_1(y), v_2 + F_2(y - w_2)]$

- Either we do not use the 2\textsuperscript{nd} item or use the 2\textsuperscript{nd} item at least once
## Weighted Knapsack Problem

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Complexity = ??**
Problem

\[
\begin{array}{c}
\text{buy} & \text{hold} & \text{hold} & \text{hold} & \text{sell} \\
\text{hold} & \text{hold} & \text{hold} & \text{hold} & \text{hold} \\
\end{array}
\]
Problem

- Decision Variables
- State Variables
Problem
Problem

• Tangent Vs The whole curve

• Looking backward from your goal
Problem

- Any sub-policy of an optimum policy must itself be optimum policy with regard to the initial and terminal states of the sub-policy
Interval Scheduling Problem

Earliest start does not work

Shortest does not work

Min-Conflict does not work