Dynamic Programming II

Inge Li Gørtz

KT section 6.4 and 6.6

Thank you to Kevin Wayne for inspiration to slides

Dynamic Programming

- Optimal substructure
- Last time
  - Weighted interval scheduling
- Today
  - Knapsack
  - Sequence alignment

Subset Sum and Knapsack

Subset Sum

- Subset Sum
  - Given *n* items $\{1,\ldots,n\}$
  - Item i has weight w_i
  - Bound W
  - Goal: Select maximum weight subset S of items so that

$$\sum_{i \in S} w_i \leq W$$

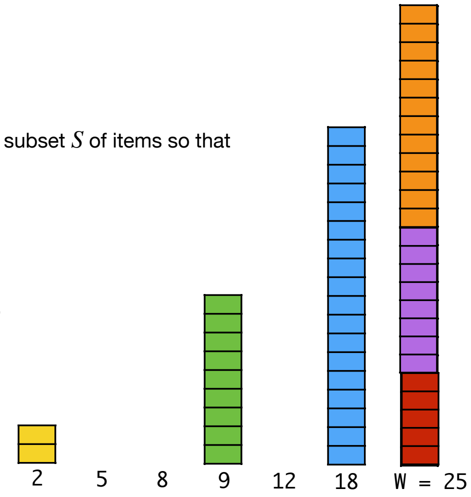
- Example
  - {2, 5, 8, 9, 12, 18} and W = 25.
  - Solution: 5 + 8 + 12 = 25.

Subset Sum

- Subset Sum
  - Given n items $\{1,\ldots,n\}$
  - Item i has weight w_i
  - Bound W
  - Goal: Select maximum weight subset S of items so that

$$\sum_{i \in S} w_i \leq W$$

- Example
  - {2, 5, 8, 9, 12, 18} and W = 25.
  - Solution: 5 + 8 + 12 = 25.



Subset Sum

- $\mathcal{O}$ = optimal solution
- Consider element n.
  - Either in $\mathcal{O}$ or not.
    - $n \notin \mathcal{O}$: Optimal solution using items $\{1,\ldots,n-1\}$ is equal to $\mathcal{O}$.
    - $n \in \mathcal{O}$: Value of $\mathcal{O} = w_n$ + weight of optimal solution on $\{1,\ldots,n-1\}$ with capacity $W - w_n$.
- Recurrence
  - $\text{OPT}(i, w)$ = optimal solution on $\{1,\ldots,i\}$ with capacity w.
  - From above:

$$\text{OPT}(n, W) = \max(\text{OPT}(n-1, W), w_n + \text{OPT}(n-1, W - w_n))$$

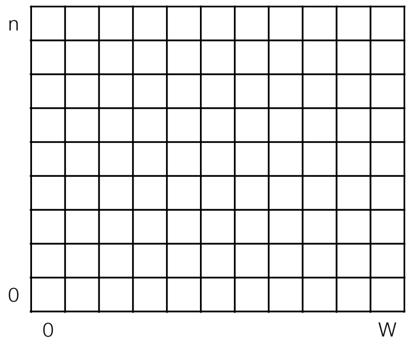
  - If $w_n > W$:

$$\text{OPT}(n, W) = \text{OPT}(n-1, W)$$

Subset Sum

- Recurrence:

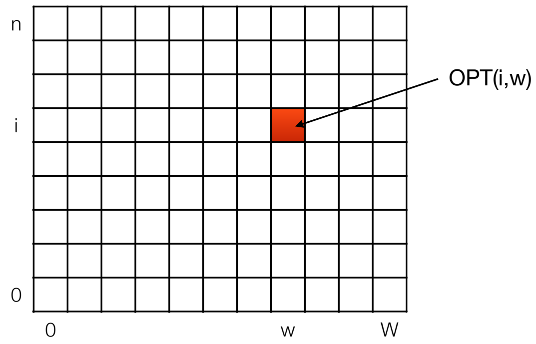
$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$



Subset Sum

- Recurrence:

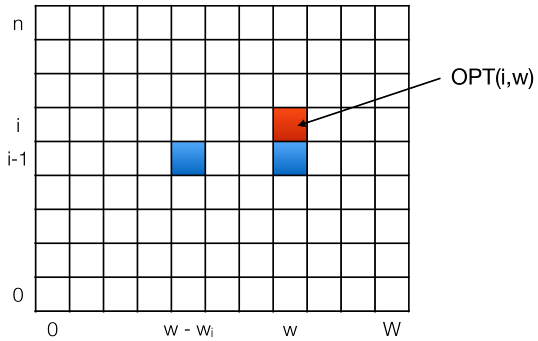
$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$



Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$



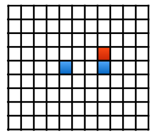
Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

```
Array M[0…n][0…W]
Initialize M[0][w] = 0 for each w = 0,1,…,W
Subset-Sum(n,W)

Subset-Sum(i,w)
  if M[i][w] empty
    if w < wᵢ
      M[i][w] = Subset-Sum(i-1,w)
    else
      M[i][w] = max(Subset-Sum(i-1,w), wᵢ +
      Subsetsum(i-1,w-wᵢ))
  return M[i][w]
```



Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

0   1 + 0

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													?
5	3													?
2	2													?
1	1													1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

1   2 + 1

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													?
5	3													?
2	2													3
1	1											1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\mathsf{OPT}(i,w) = \begin{cases} \mathsf{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\mathsf{OPT}(i-1,w), w_i + \mathsf{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4													?
5	3													?
2	2								?					3
1	1						1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\mathsf{OPT}(i,w) = \begin{cases} \mathsf{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\mathsf{OPT}(i-1,w), w_i + \mathsf{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

3 5 + 3

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4													?
5	3													8
2	2								3					3
1	1						1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\mathsf{OPT}(i,w) = \begin{cases} \mathsf{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\mathsf{OPT}(i-1,w), w_i + \mathsf{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4													?
5	3					?								8
2	2								3					3
1	1						1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\mathsf{OPT}(i,w) = \begin{cases} \mathsf{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\mathsf{OPT}(i-1,w), w_i + \mathsf{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4													?
5	3					?								8
2	2					?			3					3
1	1					?	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													?
5	3					?								8
2	2					?			3					3
1	1			?		1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													?
5	3					?								8
2	2					3			3					3
1	1			1		1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													?
5	3					3								8
2	2					3			3					3
1	1			1		1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\text{OPT}(i, w) = \begin{cases} \text{OPT}(i-1, w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w - w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

		0	1	2	3	4	5	6	7	8	9	10	11	12
9	5													?
8	4													11
5	3					3								8
2	2					3			3					3
1	1			1		1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4				?									11
5	3					3								8
2	2					3			3					3
1	1			1		1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													?
8	4				3									11
5	3				3	3								8
2	2				3	3			3					3
1	1		1	1	1	1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

9	5													12
8	4				3									11
5	3				3	3								8
2	2				3	3			3					3
1	1		1	1	1	1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Example
  - {1, 2, 5, 8, 9} and W = 12

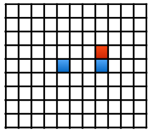
9	5													12
8	4				3									11
5	3				3	3								8
2	2				3	3			3					3
1	1		1	1	1	1	1		1			1		1
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9	10	11	12

Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

```
Subset-Sum(n,W)
  Array M[0…n][0…W]
  Initialize M[0][w] = 0 for each w = 0,1,…,W
  for i = 1 to n
    for w = 0 to W
      if w < wi
        M[i][w] = M[i-1][w]
      else
        M[i][w] = max(M[i-1][w], wi + M[i-1][w-wi])
  return M[n,W]
```

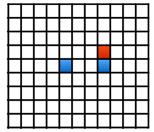


Subset Sum

- Recurrence:

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), w_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Running time:
  - Number of subproblems = nW
  - Constant time on each entry $\Rightarrow O(nW)$
  - *Pseudo-polynomial time.*
    - Not polynomial in input size:
      - whole input can be described in O(n log n + n log w) bits, where w is the maximum weight (including W) in the instance.

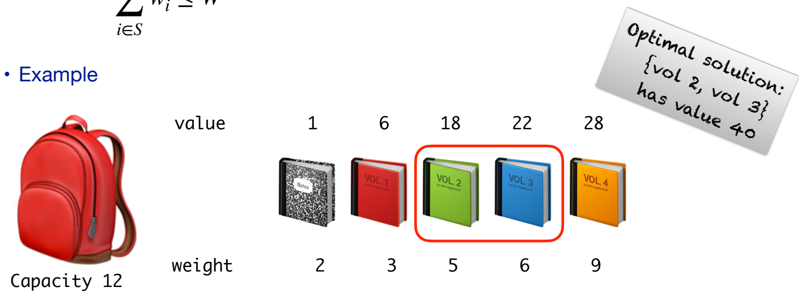


Knapsack

- Knapsack
  - Given n items $\{1,\ldots,n\}$
  - Item i has weight w_i and value v_i
  - Bound W
  - Goal: Select maximum *value* subset S of items so that

$$\sum_{i \in S} w_i \leq W$$

- Example



Knapsack

Knapsack

- $\mathcal{O}$ = optimal solution
- Consider element n.
  - Either in $\mathcal{O}$ or not.
    - $n \notin \mathcal{O}$: Optimal solution using items $\{1,\ldots,n-1\}$ is equal to $\mathcal{O}$.
    - $n \in \mathcal{O}$: Value of $\mathcal{O}$ = v_n + value on optimal solution on $\{1,\ldots,n-1\}$ with capacity $W - w_n$.

- Recurrence
  - OPT(i, w) = optimal solution on $\{1,\ldots,i\}$ with capacity w.

$$\text{OPT}(i,w) = \begin{cases} \text{OPT}(i-1,w) & \text{if } w < w_i \\ \max(\text{OPT}(i-1,w), v_i + \text{OPT}(i-1,w-w_i)) & \text{otherwise} \end{cases}$$

- Running time $O(nW)$

Dynamic programming

- **First formulate the problem recursively.**
  - Describe the *problem* recursively in a clear and precise way.
  - Give a recursive formula for the problem.

- **Bottom-up**
  - Identify all the subproblems.
  - Choose a memoization data structure.
  - Identify dependencies.
  - Find a good evaluation order.

- **Top-down**
  - Identify all the subproblems.
  - Choose a memoization data structure.
  - Identify base cases.
  - Remember to save results and check before computing.

Sequence Alignment

Sequence alignment

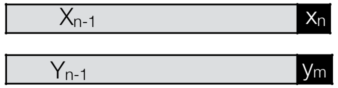
- How similar are ACAAGTC and CATGT.
- Align them such that
  - all items occurs in at most one pair.
  - no crossing pairs.
- Cost of alignment
  - gap penalty δ
  - mismatch cost for each pair of letters α(p,q).
- Goal: find minimum cost alignment.
- Input to problem: 2 strings X nd Y, gap penalty δ, and penalty matrix α(p,q).

```
A C A A G T C        A C A A - G T C
- C A T G T -        - C A - T G T -

1 mismatch, 2 gaps    0 mismatches, 4 gaps
```

Sequence Alignment

- Subproblem property.



- In the optimal alignment either:
  - x_n and y_m are aligned.
    - OPT = price of aligning x_n and y_m + minimum cost of aligning X_{i-1} and Y_{j-1}.
  - x_n and y_m are not aligned.
    - Either x_n and y_m (or both) is unaligned in OPT. Why?
    - OPT = δ + min(min cost of aligning X_{n-1} and Y_m, min cost of aligning X_n and Y_{m-1})

Sequence Alignment

- Subproblem property.

- $SA(X_i,Y_j)$ = min cost of aligning strings X[1…i] and Y[1…j].

- Case 1. Align x_i and y_j.
  - Pay mismatch cost for x_i and y_j + min cost of aligning X_{i-1} and Y_{j-1}.
- Case 2. Leave x_i unaligned.
  - Pay gap cost + min cost of aligning X_{i-1} and Y_j.
- Case 3. Leave y_j unaligned.
  - Pay gap cost + min cost of aligning X_i and Y_{j-1}.

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$SA(X_5, Y_3)$

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$SA(X_5, Y_3)$
Depends on ?

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j)=\begin{cases} j\delta & \text{if } i=0\\ i\delta & \text{if } j=0\\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$SA(X_5, Y_3)$

Depends on ?

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j)=\begin{cases} j\delta & \text{if } i=0\\ i\delta & \text{if } j=0\\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j)=\begin{cases} j\delta & \text{if } i=0\\ i\delta & \text{if } j=0\\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

min(1+0, 1+1, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j)=\begin{cases} j\delta & \text{if } i=0\\ i\delta & \text{if } j=0\\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

min(1+0, 1+1, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(0+1, 1+2, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(0+1, 1+2, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+2, 1+3, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+2, 1+3, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

min(1+3, 1+4, 1+2)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

min(1+3, 1+4, 1+2)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3			
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

min(2+4, 1+5, 1+3)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4		
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
A	2	1	2	1	2	3	4	5
T	3	2	3	2	3	3	3	4
G	4	3	4	3	4	3	4	5
T	5	4	5	4	5	4	3	4

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

```
SA(X[1…m],Y[1…n],δ,A){
  for i=0 to m
    M[i,0] := iδ

  for j=0 to n
    M[0,j] := jδ

  for i=1 to m
    for j = 1 to n
      M[i,j] := min{ A[i,j] + M[i-1,j-1],
                     δ + M[i-1,j],
                     δ + M[i,j-1]}

  Return M[m,n]
}
```

- Time: Θ(mn)
- Space: Θ(mn)

Sequence alignment: Finding the solution

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

δ = 1

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
A	2	1	2	1	2	3	4	5
T	3	2	3	2	3	3	3	4
G	4	3	4	3	4	3	4	5
T	5	4	5	4	5	4	3	4

		A	C	A	A	G	T	C
		←	←	←	←	←	←	←
C	↑	↖	↖	←	←	←	←	↖
A	↑	↖	↖	↖	↖	←	←	←
T	↑	↑	↑	↑	↖	↖	↖	←
G	↑	↑	↖	↑	↖	↖	↖	↖
T	↑	↑	↑	↑	↖	↑	↖	←

Sequence alignment

- Use dynamic programming to compute an optimal alignment.
  - Time: Θ(mn)
  - Space: Θ(mn)
- Find actual alignment by backtracking (or saving information in another matrix).
- Linear space?
  - Easy to compute value (save last and current row)
  - How to compute alignment? Hirschberg. (not part of the curriculum).