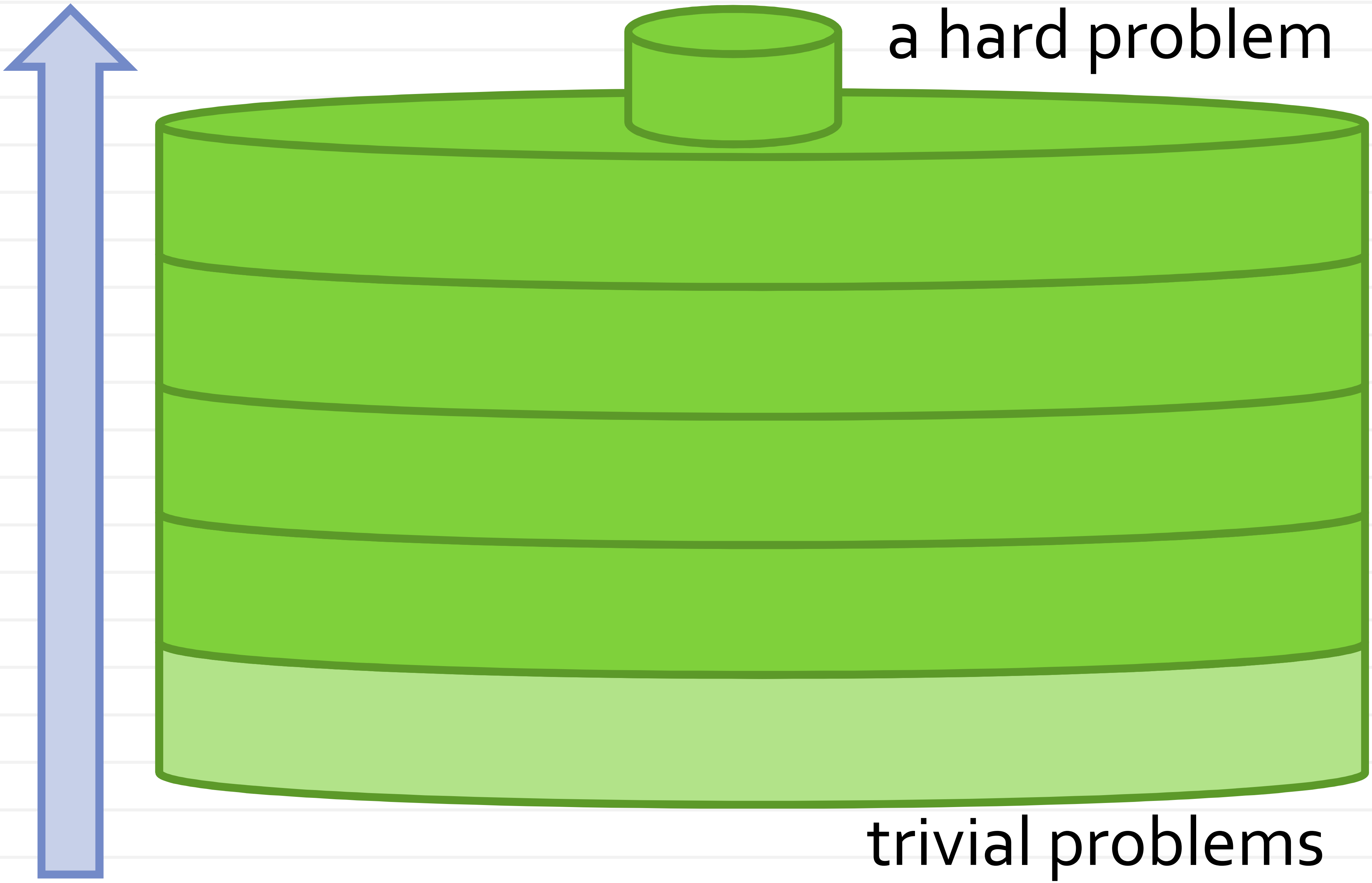
Lecture 3: Dynamic programming

Why *dynamic?*

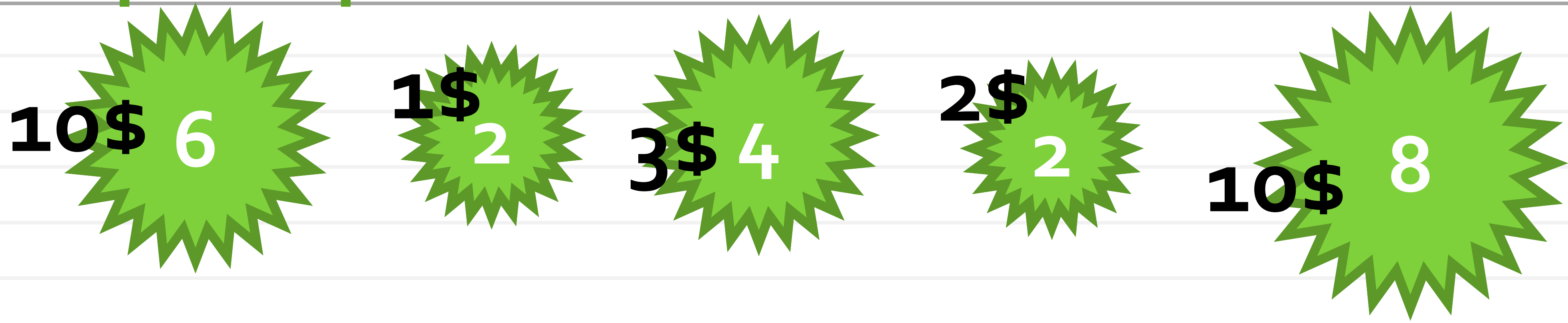
“...I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics... What title, what name, could I choose? I thought <...> lets take a word that has an absolutely precise meaning, namely *dynamic*, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense...Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities."

R. Bellman, Eye of the Hurricane: An Autobiography (1984)

Dynamic programming idea



Knapsack problem



$$\max\ c^T x$$
$$\text{s.t.: } s^T x \le K$$
$$x_i \in \{0;1\}$$

c = [10 1 3 2 10]
s = [6 2 4 2 8]

DP for knapsack

$$O(K,N) = \boxed{\begin{aligned} &\max \sum_{i=1}^{N} c_i x_i \\ &\text{s.t.} \sum_{i=1}^{N} s_i x_i \le K \\ &x_i \in \{0;1\} \end{aligned}}$$

$k \le K$

$n \le N$

$$O(k,n) = \boxed{\begin{aligned} &\max \sum_{i=1}^{n} c_i x_i \\ &\text{s.t.} \sum_{i=1}^{n} s_i x_i \le k \\ &x_i \in \{0;1\} \end{aligned}}$$

$$O(k,0) = 0$$

DP for knapsack

$k \le K$

$n \le N$

$$O(k,n) = \boxed{\begin{aligned} &\max \sum_{i=1}^{n} c_i x_i \\ &\text{s.t. } \sum_{i=1}^{n} s_i x_i \le k \\ &x_i \in \{0;1\} \end{aligned}}$$

$$O(k,0) = 0$$

Key formula (Bellman equations):

$n \ge 1$

$$O(k,n) = \begin{cases} O(k,n-1), & s_n > k \\ \max(O(k,n-1), O(k-s_n,n-1)+c_n), & s_n \le k \end{cases}$$

Dynamic programming: computation

	n=0	1	2	3	4	5
k=0	0					
1	0					
2	0					
3	0					
4	0					
5	0					
6	0					
7	0					
8	0					

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} s_i x_i \leq k$$

$$x_i \in \{0;1\}$$

c = [10 1 3 2 10]

s = [6 2 4 2 8]

Dynamic programming: computation

	n=0	1	2	3	4	5
k=0	0	0				
1	0	0				
2	0	0				
3	0	0				
4	0	0				
5	0	0				
6	0					
7	0					
8	0					

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} s_i x_i \le k$$

$$x_i \in \{0;1\}$$

c = [10 1 3 2 10]

s = [6 2 4 2 8]

Dynamic programming: computation

	n=0	1	2	3	4	5
k=0	0	0				
1	0	0				
2	0	0				
3	0	0				
4	0	0				
5	0	0				
6	0	10				
7	0	10				
8	0	10				

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} s_i x_i \le k$$

$$x_i \in \{0;1\}$$

$c = [10\ 1\ \ 3\ \ 2\ 10]$

$s = [\ 6\ \ 2\ \ 4\ \ 2\ \ 8\ ]$

Dynamic programming: computation

	n=0	1	2	3	4	5
k=0	0	0	0			
1	0	0	0			
2	0	0	1			
3	0	0	1			
4	0	0	1			
5	0	0	1			
6	0	10	10			
7	0	10	10			
8	0	10	11			

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} s_i x_i \leq k$$

$$x_i \in \{0;1\}$$

c = [10 1 3 2 10]

s = [6 2 4 2 8]

Dynamic programming: computation

	n=0	1	2	3	4	5
k=0	0	0	0	0		
1	0	0	0	0		
2	0	0	1	1		
3	0	0	1	1		
4	0	0	1			
5	0	0	1			
6	0	10	10			
7	0	10	10			
8	0	10	11			

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} s_i x_i \le k$$

$$x_i \in \{0; 1\}$$

c = [10 1 3 2 10]

s = [6 2 4 2 8]

Dynamic programming: computation

	n=0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	1	1	2	2
3	0	0	1	1	2	2
4	0	0	1	3	3	3
5	0	0	1	3	3	3
6	0	10	10	10	10	10
7	0	10	10	10	10	10
8	0	10	11	11	12	12

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t. } \sum_{i=1}^{n} s_i x_i \le k$$

$$x_i \in \{0; 1\}$$

$c = [10\ 1\ 3\ 2\ 10]$

$s = [6\ 2\ 4\ 2\ 8]$

Dynamic programming: backtracking

	n=0	1	2	3	4	5
k=0	**0**	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	1	1	2	2
3	0	0	1	1	2	2
4	0	0	1	3	3	3
5	0	0	1	3	3	3
6	0	**10**	**10**	**10**	10	10
7	0	10	10	10	10	10
8	0	10	11	11	**12**	**12**

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t. } \sum_{i=1}^{n} s_i x_i \leq k$$

$$x_i \in \{0; 1\}$$

c = [10 1 3 2 10]

s = [6 2 4 2 8]

Knapsack problem

$$\max\ c^T x$$
$$\text{s.t.: } s^T x \le K$$
$$x_i \in \{0; 1\}$$

$$c = [10\ \ 1\ \ 3\ \ 2\ \ 10]$$
$$s = [6\ \ 2\ \ 4\ \ 2\ \ 8]$$

Dynamic programming in MATLAB

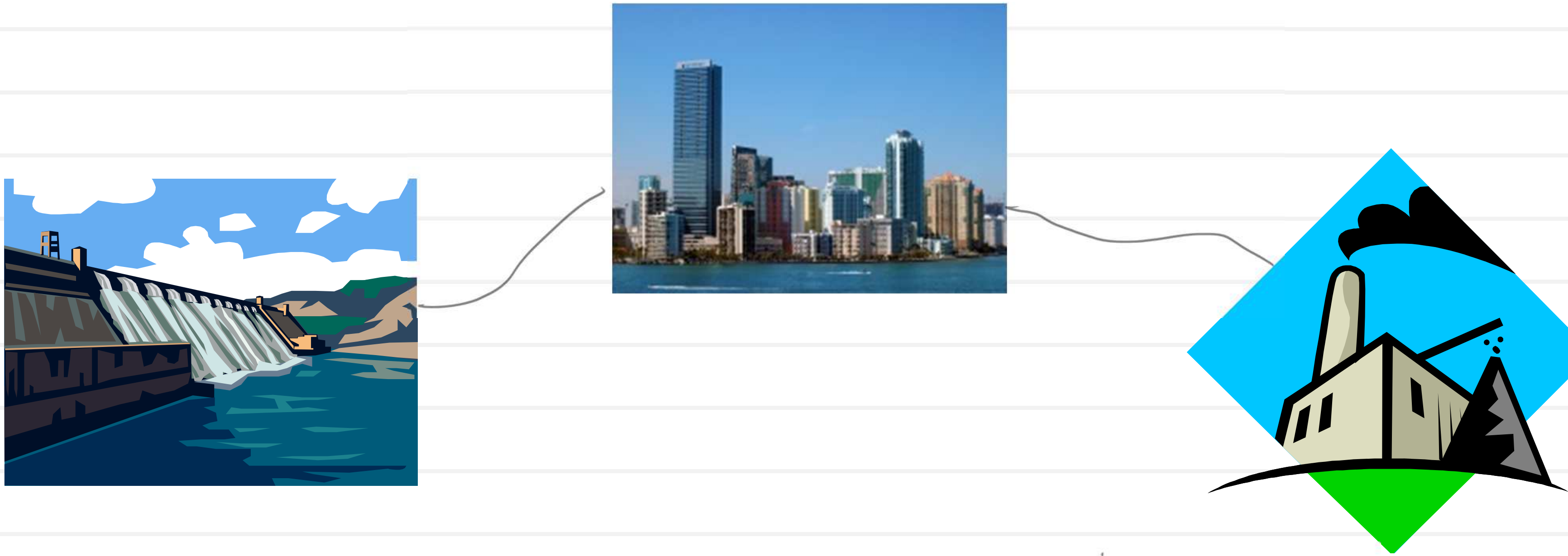
```
costs=[10 1 3 2 10];
sizes=[6 2 4 2 8];

K = 8;

table = zeros(K+1,numel(costs)+1); % table(k+1,n+1) = O(k,n)

for i=2:numel(costs)+1
  table(1:sizes(i-1),i) = table(1:sizes(i-1),i-1);
  table(sizes(i-1)+1:end,i) = max(table(sizes(i-1)+1:end,i-1),...
                        table(1:end-sizes(i-1),i-1)+costs(i-1));
end
```

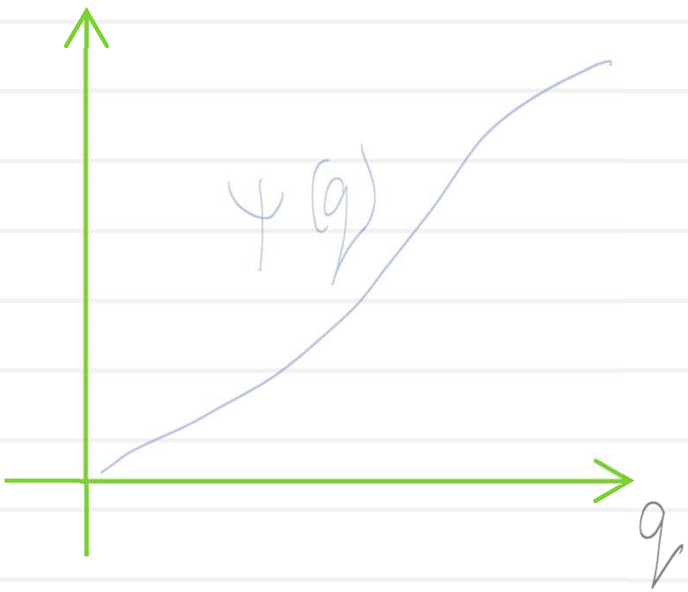
Two enabling properties for DP

- Optimal solution for the bigger problem gives optimal solutions for subproblems.
  Optimal knapsack for (K,N) also gives us several optimal knapsacks with smaller capacities / item sets

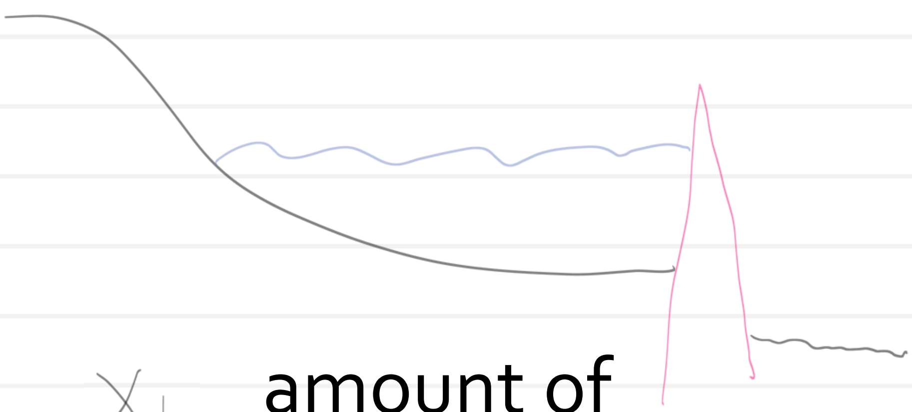
- Number of subproblems is not excessive.
  In our case, it was K*N (so that we could solve it)

Hydrothermal scheduling





Hydrothermal scheduling



$$\min_{v,q,x} \sum_{t=1}^{T} \varphi(x_t)$$

$$0 \le v_t \le V$$

$$v_t = v_{t-1} + f_t - s(v_{t-1}) - q_t$$

$$\psi(q_t) + x_t \ge d_t$$

x_t amount of "thermal" energy

$\varphi(x_t)$ cost of "thermal" energy

q_t amount of water consumed

$\psi(q_t)$ amount of "hydro" energy produced

v_t the volume of water at the end of the period t

f_t the inflow of water

$s(\cdot)$ the spillage of water

d_t power demand

v_0 initial volume of water

Hydrothermal scheduling

$$O(T) =$$

$$\min_{v,q,x} \sum_{t=1}^{T} \varphi(x_t)$$

$$0 \le v_t \le V$$

$$v_t = v_{t-1} + f_t - s(v_{t-1}) - q_t$$

$$\psi(q_t) + x_t \ge d_t$$

$$O(n,u) =$$

$$\min_{v,q,x} \sum_{t=1}^{n} \varphi(x_t)$$

$$0 \le v_t \le V$$

$$v_t = v_{t-1} + f_t - s(v_{t-1}) - q_t$$

$$\psi(q_t) + x_t \ge d_t$$

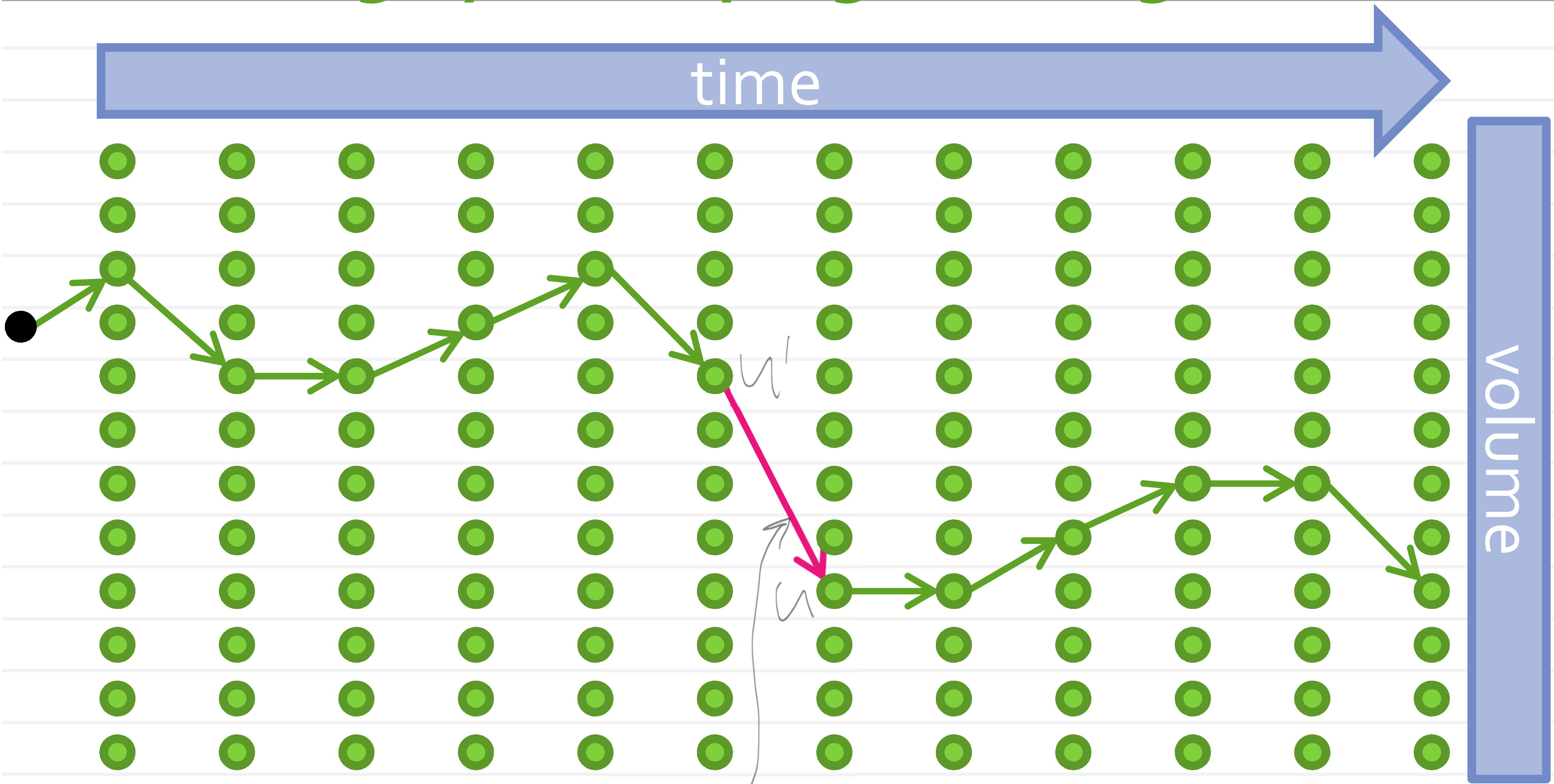
$$v_n = u$$

Bellman equation:

$$O(n,u) = \min_{u'} \Big(O(n-1, u') + \varphi\big[d_n - \psi(u' - u + f_n - s(u'))\big] \Big)$$

Key idea: discretizing the volume variable

Visualizing dynamic programming



$$\varphi\left[d_n - \psi\left(u' - u + f_n - s(u')\right)\right]$$

General framework: optimizing actions

$$\max_{a,s} \sum_{i=1}^{N} r_i(s_{i-1}, a_i) + r'_N(s_N)$$

$$\text{s.t.: } s_i = T_i(s_{i-1}, a_i), \; i = 1 \ldots N-1$$

$$s_0 = \tilde{s}$$

s_i	state after step *i*
a_i	action at step *i*
$T_i(s_{i-1}, a_i)$	transition rule *i*
$\tilde{s}$	initial state
$r_i(s_{i-1}, a_i), r'_N(s_N)$	rewards

General framework: optimizing actions

$$O(N) = \boxed{\max_{a,s} \sum_{i=1}^{N} r_i(s_{i-1}, a_i) + r'_N(s_N) \quad \text{s.t.: } s_i = T_i(s_{i-1}, a_i),\ i=1\ldots N-1, \quad s_0 = \tilde{s}}$$

$$O(n,s) \quad \boxed{\max_{a,s} \sum_{i=1}^{n} r_i(s_{i-1}, a_i) \quad \text{s.t.: } s_i = T_i(s_{i-1}, a_i), \quad s_0 = \tilde{s},\ s_n = s}$$

$$O(N) = \max_{s} \left[O(N,s) + r'_N(s) \right]$$

General framework: optimizing actions

$$O(n,s) \quad \boxed{\begin{aligned} &\max_{a,s} \sum_{i=1}^{n} r_i(s_{i-1}, a_i) \\ &\text{s.t.}: s_i = T_i(s_{i-1}, a_i) \\ &\qquad s_0 = \tilde{s},\ s_n = s \end{aligned}}$$

$$O(N) = \max_s \left[O(N,s) + r_N'(s) \right]$$

$$O(n,s) = \max_{(s',a') \in T_n^{-1}(s)} \left[O(n-1, s') + r_i(s', a') \right]$$

$$T_n^{-1}(s) = \{ (s', a') \mid T_n(s', a') = s \}$$

Part-of-speech tagging

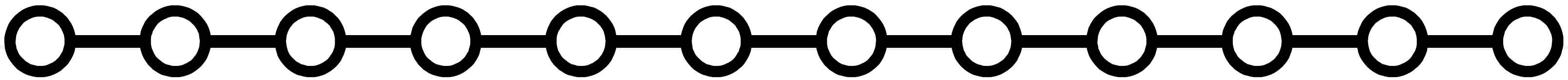
VBZ

DT/ The JJ/ quick JJ/ brown NN/ fox ~~NNS~~/ jumps IN/ over DT/ the JJ/ lazy NN/ dog

- -RRB- - Right bracket
- CD - Cardinal number
- EX - Existential there
- IN - Preposition
- JJR - Comparative adjective
- LS - List Item Marker
- NN - Singular noun
- NNP - Proper singular noun
- PDT - Predeterminer
- PRP - Personal pronoun
- RB - Adverb
- RBS - Superlative Adverb
- SYM - Symbol
- UH - Interjection
- VBD - Verb, past tense
- VBN - Verb, past participle
- VBZ - Verb, 3rd ps. sing. present
- WP - wh-pronoun
- WRB - wh-adverb
- CC - Coordinating conjunction
- DT - Determiner
- FW - Foreign word
- JJ - Adjective
- JJS - Superlative adjective
- MD - Modal
- NNS - Plural noun
- NNPS - Proper plural noun
- POS - Possesive ending
- PP$ - Possesive pronoun
- RBR - Comparative adverb
- RP - Particle
- TO - to
- VB - Verb, base form
- VBG - Verb, gerund/present participle
- VBP - Verb, non 3rd ps. sing. present
- WDT - wh-determiner
- WP$ - Possesive wh-pronoun

http://cogcomp.cs.illinois.edu/demo/pos/

Part-of-Speech tagging



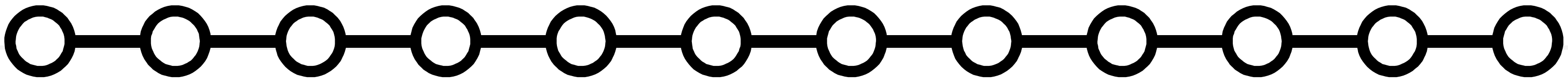
$$E(y) = \sum_{i=1}^{W} E_i(y_i) + \sum_{i=1}^{W-1} E_{i,i+1}(y_i, y_{i+1})$$

Dictionary information, capitalization, suffixes and prefixes, all of this for the preceding and subsequent words, etc.

Transition probabilities

$$O(k,t) = \min_{y_1 \dots y_k} \sum_{i=1}^{k} E_i(y_i) + \sum_{i=1}^{k-1} E_{i,i+1}(y_i, y_{i+1}) \quad \text{s.t. } y_k = t$$

Part-of-Speech tagging



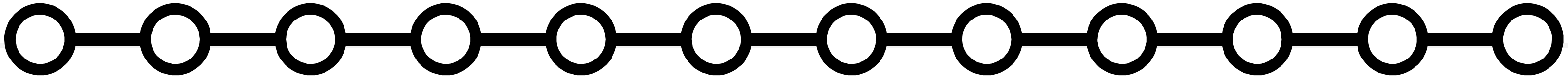
$$E(y) = \sum_{i=1}^{W} E_i(y_i) + \sum_{i=1}^{W-1} E_{i,i+1}(y_i, y_{i+1})$$

$$O(k,t) = \boxed{\begin{array}{c} \min\limits_{y_1 \dots y_k} \sum\limits_{i=1}^{k} E_i(y_i) + \sum\limits_{i=1}^{k-1} E_{i,i+1}(y_i, y_{i+1}) \\ \text{s.t. } y_k = t \end{array}}$$

Bellman equation:

$$O(k,t) = E_k(t) + \min_{t'} \left[O(k-1,t') + E_{k-1,k}(t',t) \right]$$

Part-of-Speech tagging



$$E(y) = \sum_{i=1}^{W} E_i(y_i) + \sum_{i=1}^{W-1} E_{i,i+1}(y_i, y_{i+1})$$

Dictionary information, capitalization, suffixes and prefixes, all of this for the preceding and subsequent words, etc.

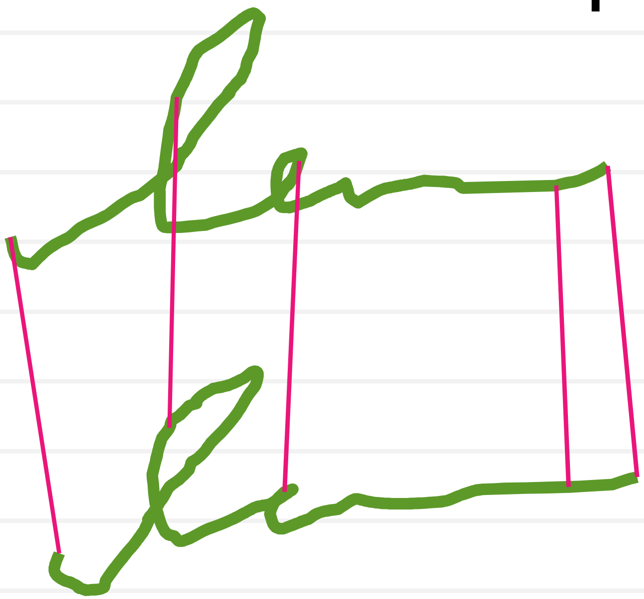
Transition probabilities

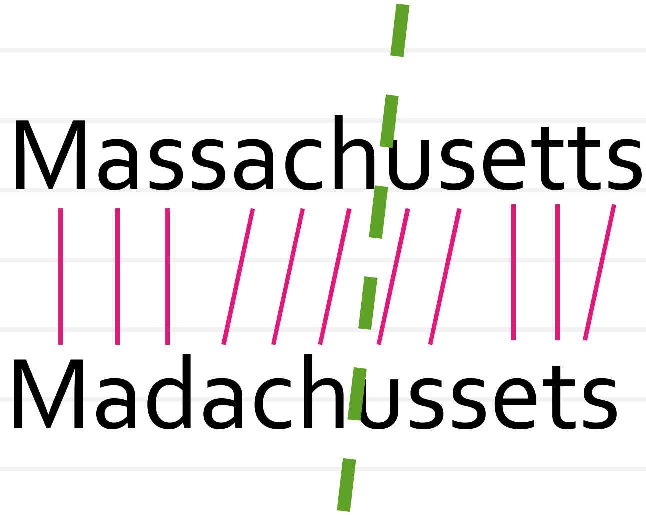
Bellman equation:

$$E(y_{1:k}, y_k = t) = E_k(t) + \min_{t'} E(y_{1:k-1}, y_{k-1} = t') + E_{k-1,k}(t', t)$$

Sequence alignment

- Linguistics/language analysis: Levenshtein edit distance
- Bioinformatics: Needleman-Wunsch algorithm
- Speech/signal processing: dynamic time warping





Sequence alignment

GGTCACTAACGCCTAAA

GGTACTAACTGCCTAGA

Distance = 2*Cost(indel) + 1*Cost(match)

$x_1, x_2 \ldots x_R \quad y_1, y_2 \ldots y_S \quad M_i \in \{0, 1, .. S\} \quad \tilde{M}_i \in \{0, 1, .. R\}$

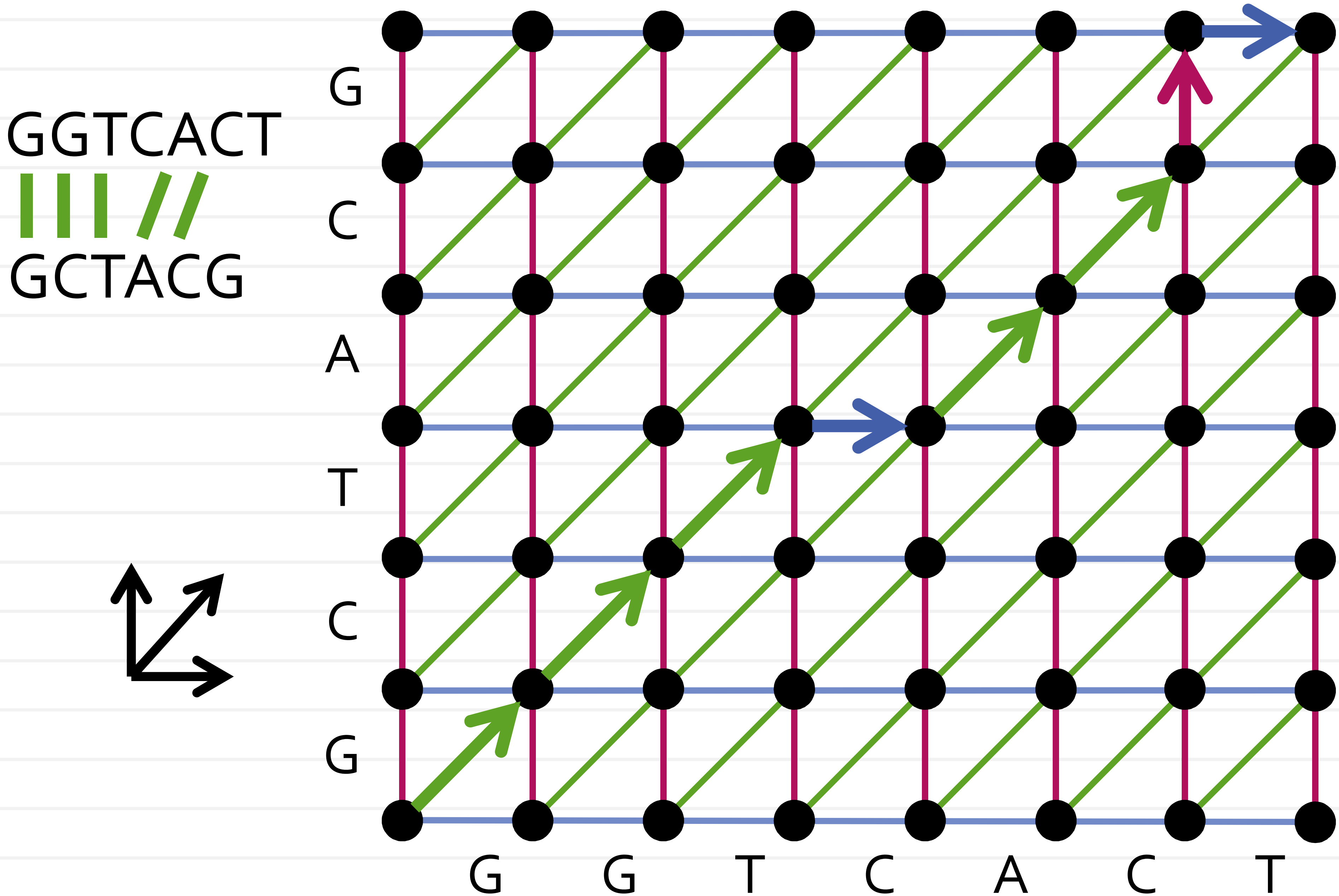
$$\min_{M, \tilde{M}} \lambda \left(\sum_{i=1}^{R} [M_i = 0] + \sum_{i=1}^{S} [\tilde{M}_i = 0] \right) + \mu \sum_{i: M_i \neq 0} d(x_i, y_{M_i})$$

$$\text{s.t.: } \forall i < i', M_i \neq 0, M_{i'} \neq 0 : M_i < M_{i'}$$

$$\forall i < i', \tilde{M}_i \neq 0, \tilde{M}_{i'} \neq 0 : \tilde{M}_i < \tilde{M}_{i'}$$

$$\forall i, M_i \neq 0, \quad \tilde{M}_{M_i} = i \qquad \forall i, \tilde{M}_i \neq 0 \; M_{\tilde{M}_i} = i$$

Sequence alignment



Sequence alignment: dynamic programming

$x_1, x_2 \ldots x_R \quad y_1, y_2 \ldots y_S \quad M_i \in \{0,1,\ldots S\} \quad \tilde{M}_i \in \{0,1,\ldots R\}$

$$O(R,S) = \min_{M,\tilde{M}} \lambda\left(\sum_{i=1}^{R}[M_i=0] + \sum_{i=1}^{S}[\tilde{M}_i=0]\right) + \mu \sum_{i:M_i\neq 0} d(x_i, y_{M_i})$$

$$\text{s.t.:}\ \forall i<i',\ M_i\neq 0,\ M_{i'}\neq 0:\ M_i < M_{i'}$$

$$\forall i<i',\ \tilde{M}_i\neq 0,\ \tilde{M}_{i'}\neq 0:\ \tilde{M}_i < \tilde{M}_{i'}$$

$$\forall i,\ M_i\neq 0,\ \tilde{M}_{M_i}=i \quad \forall i,\ \tilde{M}_i\neq 0\ M_{\tilde{M}_i}=i$$

Bellman equation:

$$O(0,s) = \lambda s$$

$$O(r,0) = \lambda r$$

$$O(r,s) = \min\big(O(r-1,s)+\lambda,\ O(r,s-1)+\lambda,\ O(r-1,s-1)+d(x_s,y_r)\big)$$

Dynamic programming

- A powerful method, when the task decomposes
- Especially useful in problems associated with sequences
- Can be applied to continuous variables via discretization
- Does not care about the shape of the terms (convexity, etc.)
- Forward pass gives the optimal value, backtracking gives the optimal variables