Lecture 4

DYNAMIC PROGRAMMING AND SEQUENCE ALIGNMENT

Sequence similarity

- Genome rearrangement problem assumed we know for each gene in species A its counterpart in species B (if exists).
  - Orthologous genes – same ancestor in evolution.
  - Paralogous gene – gene dublication.
  - Homolog = Ortholog or Paralog
- Often sequence similarity is the only way to predict whether two genes are homologs.
  - Very unlikely that same (long sequences) have evolved independently from different ancestors.
  - … except horizontal gene transfer

Sequence similarity vs. distance

- Let A and B be two strings from alphabet Σ, i.e., $A,B \in \Sigma^*$.
- Many different ways to define the *similarity* or *distance* of A and B.
- Recall Hamming distance $d_H(A,B)$.
  - Only defined when $|A|=|B|$.
- What is the simplest measure to extend Hamming distance to different length strings?
  - For many purposes it is useful if the distance is a *metric*.

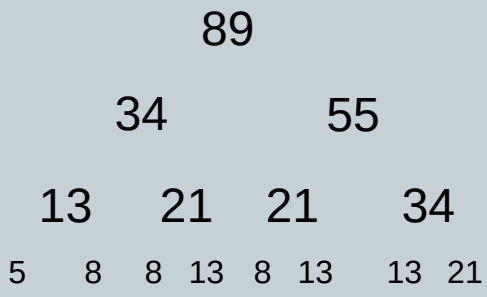
Edit distance

- The most studied distance function extending Hamming distance is *unit cost edit distance* or *Levenshtein distance*.
  - $d_L(A,B)$ is the minimum amount of single symbol *insertions*, *deletions*, and *substitutions* required to convert *A* into *B*.
  - For example, on *A=" tukholma"* and *B=" stockholm"* we have $d_L(A,B)=4$:
    - insert s, substitute u->o, insert c, delete a
    - .. or insert s, insert o, substitute u->c, delete a
    - .. or is there better sequence of edits???

```
- t u - k h o l m a
s t o c k h o l m -
```

Dynamic programming

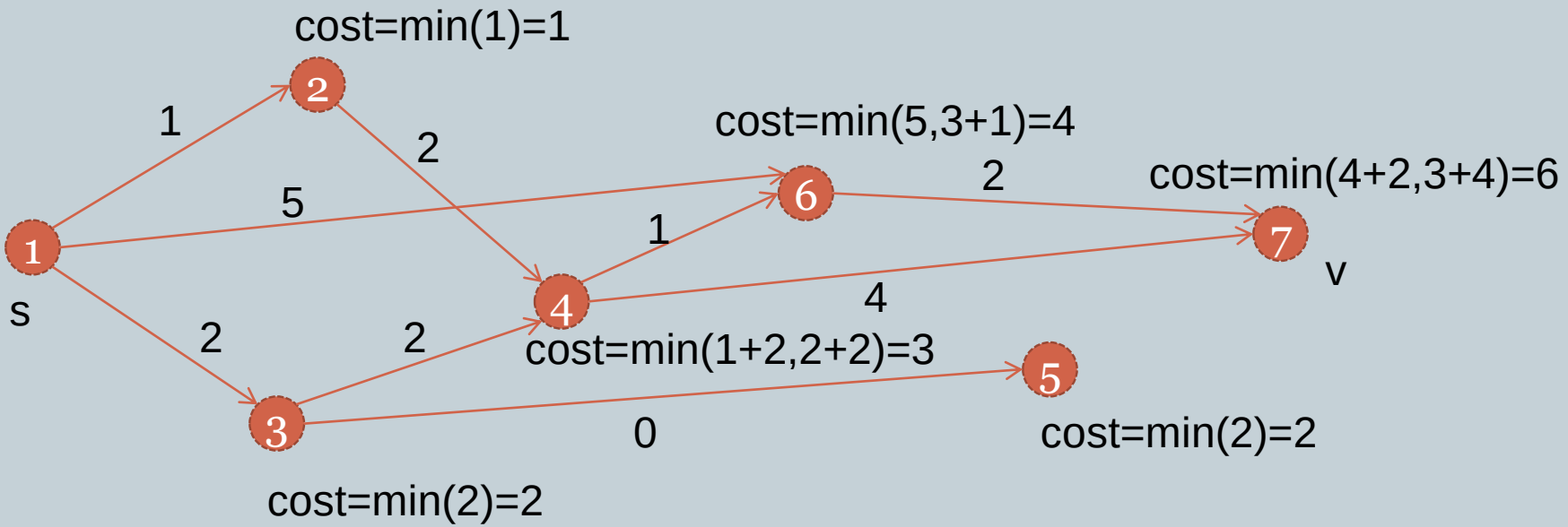
- Way to compute edit distance optimally.
- General algorithm principle:
  - Similar to *Dijkstra's shortest path algorithm*.
  - Abstract idea: Use induction to break the problem into smaller subproblems and suitable evaluation order so that subproblem solutions are available when needed.
- Concrete example, Fibonacci numbers:
  - 1,1,2,3,5,8,13,21,34,55,89,...
  - *F(i)=F(i-2)+F(i-1)* with *F(1)=1*, *F(2)=1*
  - The recursion to compute *F(i)* contains many identical subproblems.



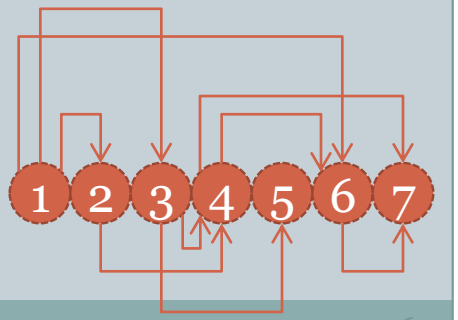
Lightest path in a DAG

DAG=directed acyclic graph

Lightest path from s to v?



Topological sort



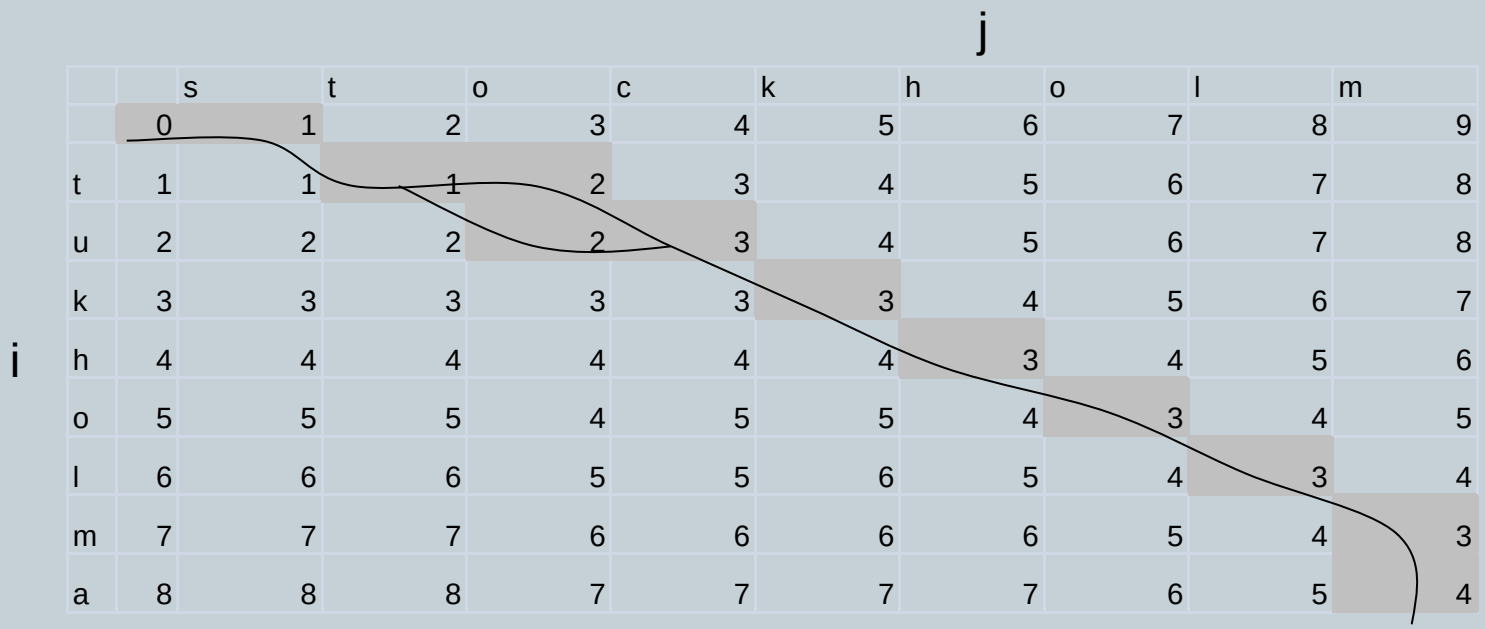
Edit distance

- Consider an optimal listing of edits to convert the prefix $a_1a_2...a_i$ of A into prefix $b_1b_2...b_j$ of B corresponding to $d_L(a_1a_2...a_i,b_1b_2...b_j)$:
  - If $a_i=b_j$ we know that $d_L(a_1a_2...a_i,b_1b_2...b_j)=d_L(a_1a_2...a_{i-1},b_1b_2...b_{j-1})$
  - Otherwise either a_i is substituted by b_j, or a_i is deleted or b_j is inserted in the optimal list of edits.
  - Hence, we have $d_L(a_1a_2...a_i,b_1b_2...b_j)=$
    $min(d_L(a_1a_2...a_{i-1},b_1b_2...b_{j-1})+(if\ a_i=b_j\ then\ 0\ else\ 1),$
    $d_L(a_1a_2...a_{i-1},b_1b_2...b_j)+1,$
    $d_L(a_1a_2...a_i,b_1b_2...b_{j-1})+1).$

Edit distance matrix D[i,j]

- Let $D[i,j]$ denote $d_L(a_1a_2...a_i,b_1b_2...b_j)$.
- Obviously $D[0,j]=j$ and $D[i,0]=i$.
- The induction from previous slide gives
  $D[i,j]=min(D[i-1,j-1]+if\ (a_i=b_j)\ then\ 0\ else\ 1,$
  $D[i-1,j]+1,D[i,j-1]+1)$.
- Matrix D can be computed row-by-row, column-by-column (or in many other evaluation orders) so that $D[i-1,j-1]$, $D[i-1,j]$, and $D[i,j-1]$ are available when computing $D[i,j]$.
- Running time to compute $D[m,n]$ is $O(mn)$.

Edit distance example



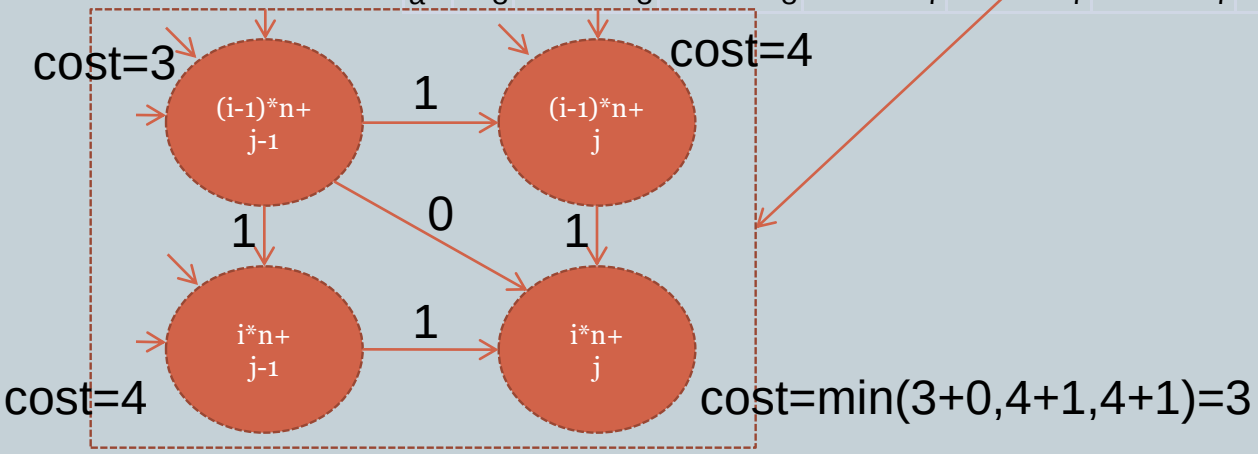
i \ j		s	t	o	c	k	h	o	l	m
	0	1	2	3	4	5	6	7	8	9
t	1	1	1	2	3	4	5	6	7	8
u	2	2	2	2	3	4	5	6	7	8
k	3	3	3	3	3	3	4	5	6	7
h	4	4	4	4	4	4	3	4	5	6
o	5	5	5	4	5	5	4	3	4	5
l	6	6	6	5	5	6	5	4	3	4
m	7	7	7	6	6	6	6	5	4	3
a	8	8	8	7	7	7	7	6	5	4

Edit distance matrix as a DAG

		s	t	o	c	k	h	o	l	m
	0	1	2	3	4	5	6	7	8	9
t	1	1	1	2	3	4	5	6	7	8
u	2	2	2	2	3	4	5	6	7	8
k	3	3	3	3	3	3	4	5	6	7
h	4	4	4	4	4	4	3	4	5	6
o	5	5	5	4	5	5	4	3	4	5
l	6	6	6	5	5	6	5	4	3	4
m	7	7	7	6	6	6	6	5	4	3
a	8	8	8	7	7	7	7	6	5	4

j

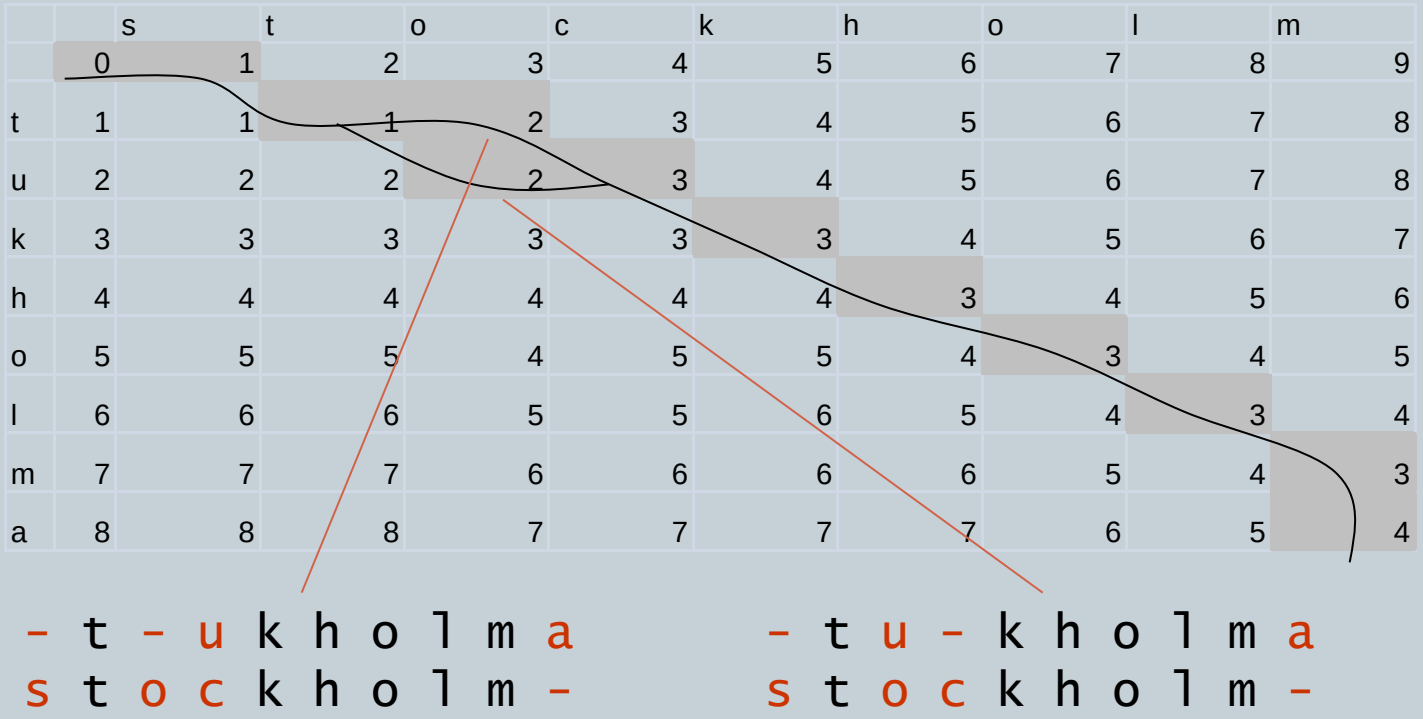
i



Finding the optimal alignment(s)

- Two options:
  - (one alignment) Store pointer to each cell telling from which cell the minimum was obtained, follow the pointers from (m,n) to (0,0) and reverse the list; or
  - (all alignments) Backtrack from (m,n) to (0,0) by checking at each cell (i,j) on the path whether the value D[i,j] could have been obtained from cell (i,j-1), (i-1,j-1), or (i-1,j). Explore all directions.
    - All three directions possible.
    - Exponential number of optimal paths in the worst case.

Edit distance example



		s	t	o	c	k	h	o	l	m
	0	1	2	3	4	5	6	7	8	9
t	1	1	1	2	3	4	5	6	7	8
u	2	2	2	2	3	4	5	6	7	8
k	3	3	3	3	3	3	4	5	6	7
h	4	4	4	4	4	4	3	4	5	6
o	5	5	5	4	5	5	4	3	4	5
l	6	6	6	5	5	6	5	4	3	4
m	7	7	7	6	6	6	6	5	4	3
a	8	8	8	7	7	7	7	6	5	4

```
- t - u k h o l m a
s t o c k h o l m -
```

```
- t u - k h o l m a
s t o c k h o l m -
```

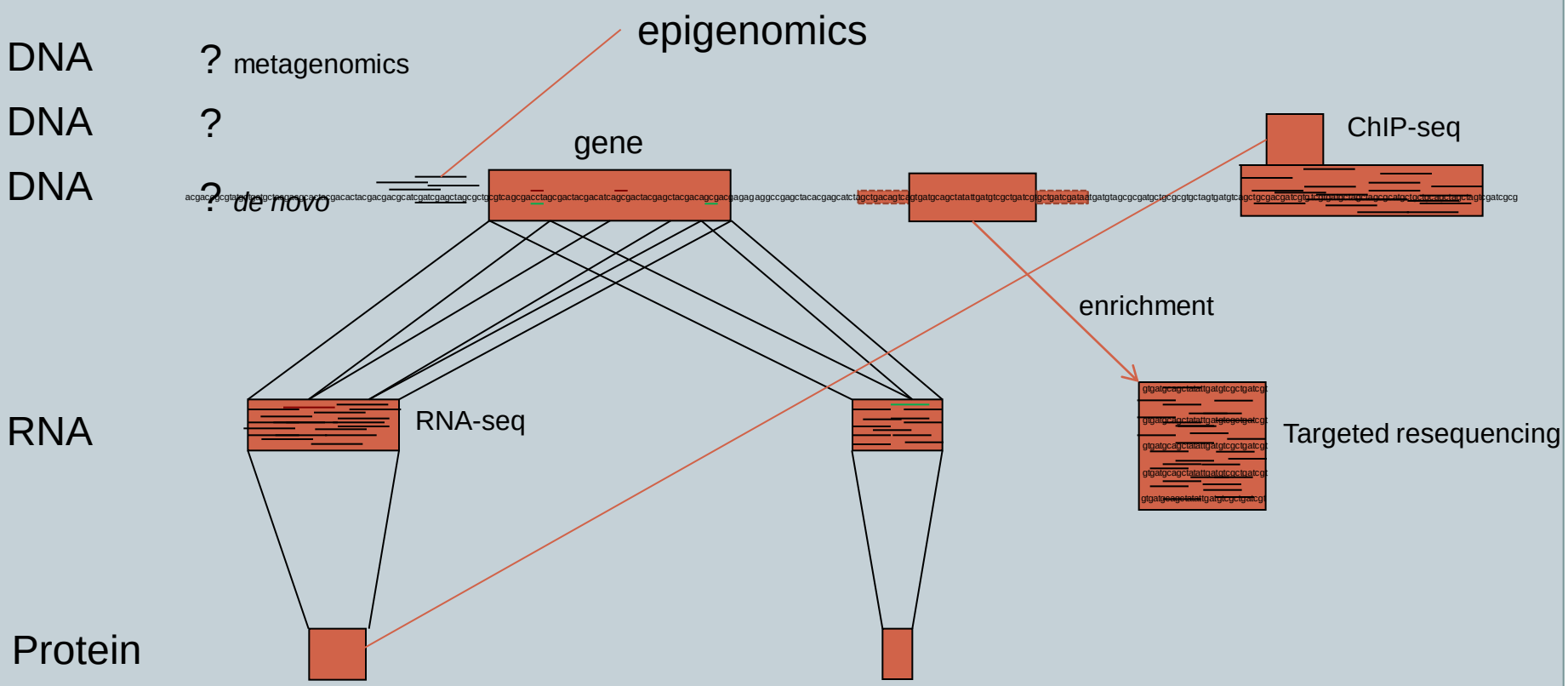
Searching homologs with edit distance?

- Take DNA sequences A and B of two genes suspected to be homologs.
- Edit distance of A and B can be *huge* even if A and B are true homologs.
  - One reason is *silent mutations* that alter DNA sequence so that the codons still encode the same amino acids.
  - In principle, A and B can differ in almost every third nucleotide.
- Better compare protein sequences.
  - Some substitutions are more likely than the others…
  - Lot of tuning needed to use proper weights for operations.

Other applications in bioinformatics

- High-throughput next-generation sequencing (NGS) has raised again the issue of using edit distance.
  - Short DNA *reads* (50-1000 bp) a.k.a. *patterns* are measured from e.g. cells of a patient.
  - The reads are aligned against the reference genome.
    - Typically only SNPs and measurement errors need to be taken into account.
    - The occurrence of the read in the reference genome can be determined by finding the substring of the genome whose edit distance (or Hamming distance) to the read is minimum.
    - Approximate string matching problem.

NGS-atlas: RNA-seq, ChIP-seq, (targeted) resequencing, *de novo* sequencing, metagenomics



Approximate string matching with d_H

- *k-mismatches problem*: Search all occurrences O of pattern $P[1,m]$ in text $T[1,n]$ such that P differs in at most k positions from the occurrence substring:
  - More formally: $j \in O$ is a k-mismatch occurrence position of P in T if and only if $d_H(P,T[j,j+m-1]) \leq k$, where $d_H(A,B)=|\{ i : A[i] \neq B[i]\}|$.
  - Compare to the TotalDistance()-computation in the exercises.
  - Naive algorithm:
    - Compare P against each $T[j,j+m-1]$ but skip as soon as k+1 mismatches are encountered.
    - Expected linear time!

Approximate string matching with d_L

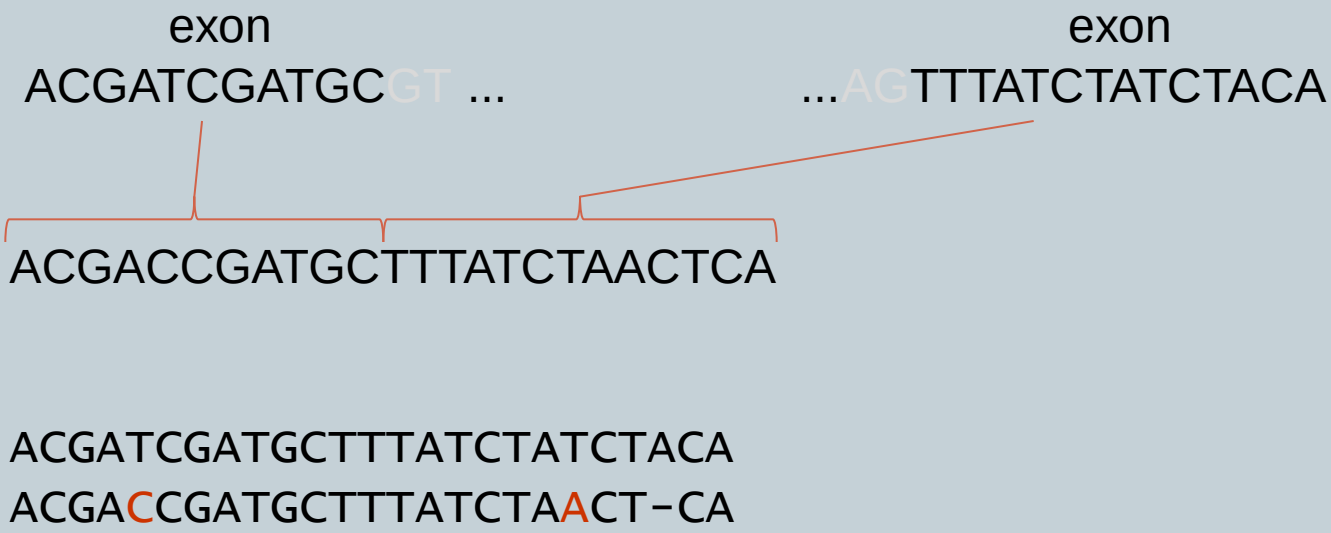
- *k-errors problem* is the approximate string matching problem with edit distance:
  - More formally: $j \in O$ is a k-errors occurrence (end)position of P in T if and only if $d_L(P,T[j',j]) \leq k$ for some j'.
- Can be solved with the "zero the first row trick":
  - $D[0,j]=0$ for all j.
  - Otherwise the computation is identical to edit distance computation using matrix D.
  - Intuition: $D[i,j]$ then equals the minimum number of edits to convert $P[1,i]$ into *some suffix of* $T[1,j]$.
  - If $D[m,j] \leq k$, then P can be converted to some substring $T[j',j]$ with at most k edit operations.

NGS atlas and approximate string matching 1/3

- Aligning reads from ChIP-seq and targeted resequencing works using basic approximate string matching, but…
  - Tens of millions of reads, spead is an issue.
  - Reference genome can be preprocessed to speed up search:
    - Suffix tree alike techniques work, but…
      - Suffix tree of human genome takes 50-200 GB!
      - More space-efficient index structures have been developed (e.g. based on *Burrows-Wheeler transform*) that drop the space to ~3 GB.

NGS atlas and approximate string matching 2/3

- Reads from RNA-seq need more advanced alignment:
  - Read can span two exons.
  - Next week exercises study this problem.

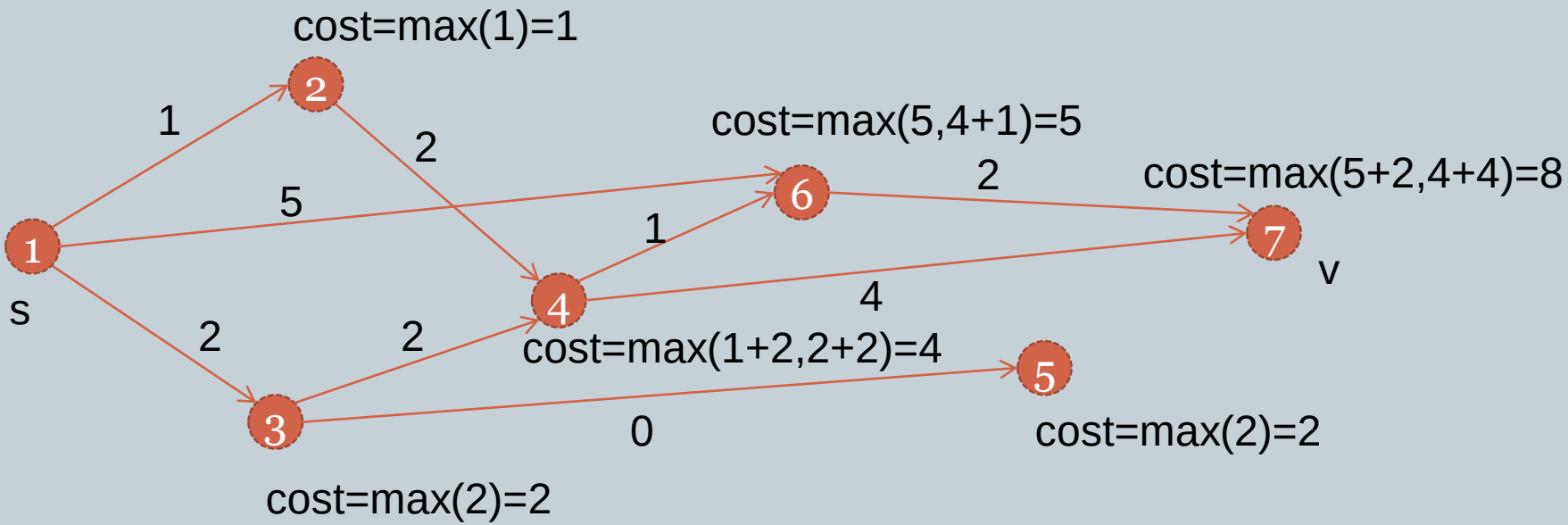


NGS atlas and approximate string matching 3/3

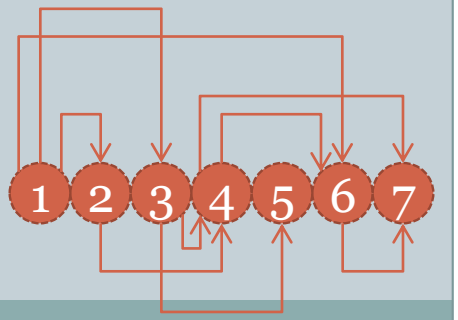
- *de novo* sequencing and metagenomics are much harder since there is no reference genome.
  - Shortest approximate superstring (exercise 3.4).
  - How to modify edit distance computation for overlaps?
    - Next week exercise.

Variations of the theme

Heaviest path from s to v?



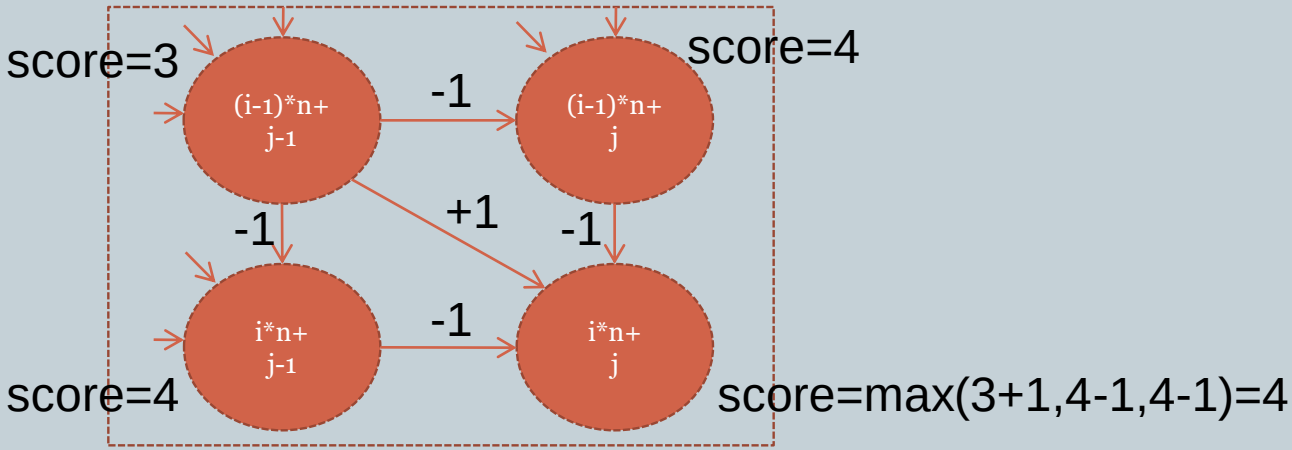
Topological sort



Heaviest paths in sequence aligment

- Consider the DAG of edit distance matrix.
- Turn minimization into maximization.
- Give *score* $\delta(a_i,b_j)$ for diagonal edges.
- Give score $\delta(a_i,-)$ for vertical edges.
- Give score $\delta(-,b_j)$ for horizontal edges.
- Then heaviest path in the DAG corresponds to the *global alignment* with highest score.
  - Typically $\delta(a_i,b_j)=1$ if $a_i=b_j$ otherwise $\delta(a_i,b_j)=-\mu$.
  - Typically $\delta(a_i,-)=\delta(-,b_j)=-\sigma$.

Global alignment DAG and recurrence

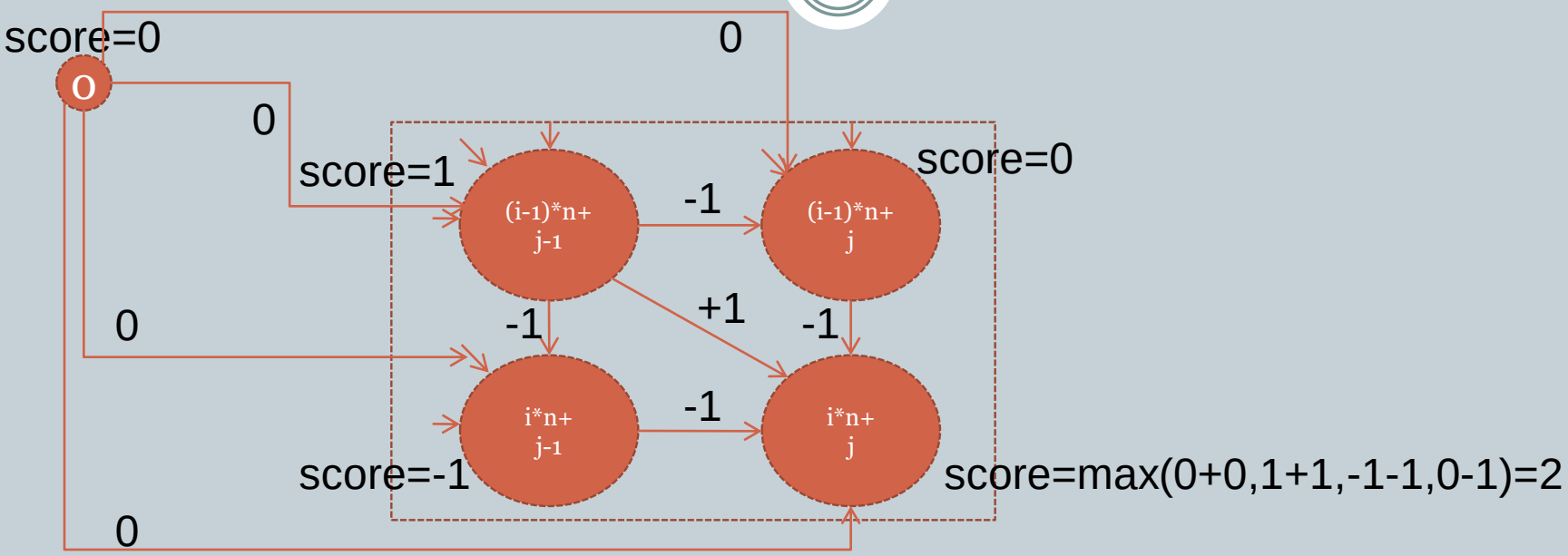


- $S[i,j]=max(S[i-1,j-1]+\delta(a_i,b_j),$
  $S[i-1,j]+\delta(a_i,-),S[i,j-1]+\delta(-,b_j))$.

Heaviest *local* paths in sequence aligment

- Consider the heaviest path DAG corresponding to global alignment with highest score.
- How to find heaviest subpaths (local path)?
- Defining that empty path has score 0, it is enough to search for subpaths (local paths) with weight greater than 0.
  - No heaviest path can have a prefix with negative score.
    - Add an edge with score 0 from node 0 to all nodes i*n+j.

Local alignment DAG and recurrence



- $S[i,j]=max(0,S[i-1,j-1]+\delta(a_i,b_j),$
  $S[i-1,j]+\delta(a_i,-),S[i,j-1]+\delta(-,b_j))$.

Longest Common Subsequence (LCS)

- Global alignment with
  - $\delta(a_i,b_j)=1$ when $a_i=b_j$ and otherwise $\delta(a_i,b_j)=-\infty$, and
  - $\delta(a_i,-)=\delta(-,b_j)=0$,

  gives the length of the longest common subsequence C of A and B:
  - Longest sequence C that can be obtained by deleting 0 or more symbols from A and also by deleting 0 or more symbols from B.

    AACGCATACGG ACGACTGATCG

    AGCTACG

  - Connection: $d_{ID}(A,B)=m+n-2*|LCS(A,B)|$, where $d_{ID}(A,B)$ is the edit distance with substitution cost ∞.

Study group assignments

MONDAY 3.10. 12-14 B222

Group 1 (random assignment at lecture)

- Small parsimony problem:
  - Dynamic programming on fixed phylogenetic tree.
  - J & P pages 368-373.
  - (copies shared at lecture)
- At study group, simulate the algorithm with some example.

Group 2 (random assignment at lecture)

- RNA secondary structure prediction:
  - Basic dynamic programming formulation.
  - See http://www.nature.com/nbt/journal/v22/n11/abs/nbt1104-1457.html
- At study group, give an example of RNA secondary structure, how the recurrence is derived for its computation, and how the recurrence is evaluated.

Group 3 (random assignment at lecture)

- Gene prediction by spliced alignment:
  - Application/extension of heaviest path on a DAG.
  - J & P pages 203-207.
  - (copies shared at lecture)
- At study group, explain the idea visually and explain how the recurrences are derived. What is the running time of the algorithm?