Sequence Alignment

Data Structures and Algorithms
Andrei Bulatov

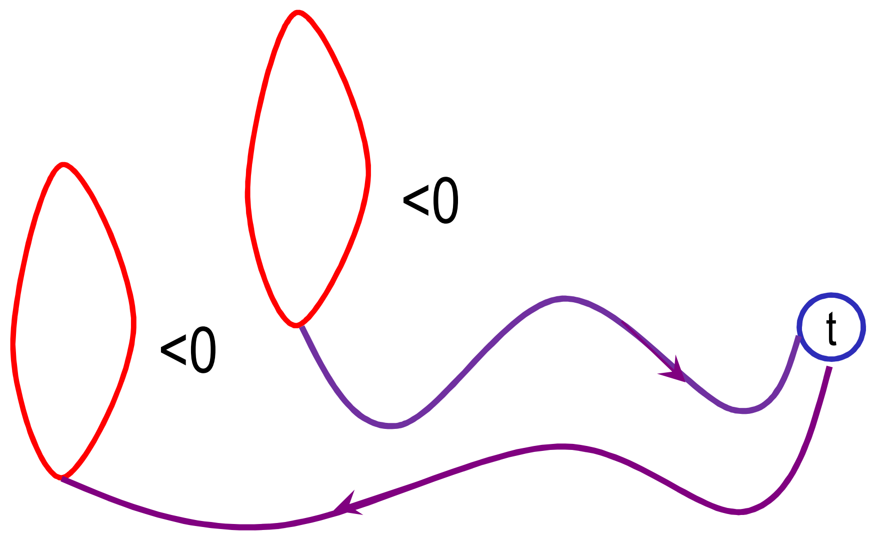
Shortest Path: Finding Negative Cycles

Two questions:

- how to decide if there is a negative cycle?
- how to find one?

Lemma

It suffices to find negative cycles C such that t can be reached from C

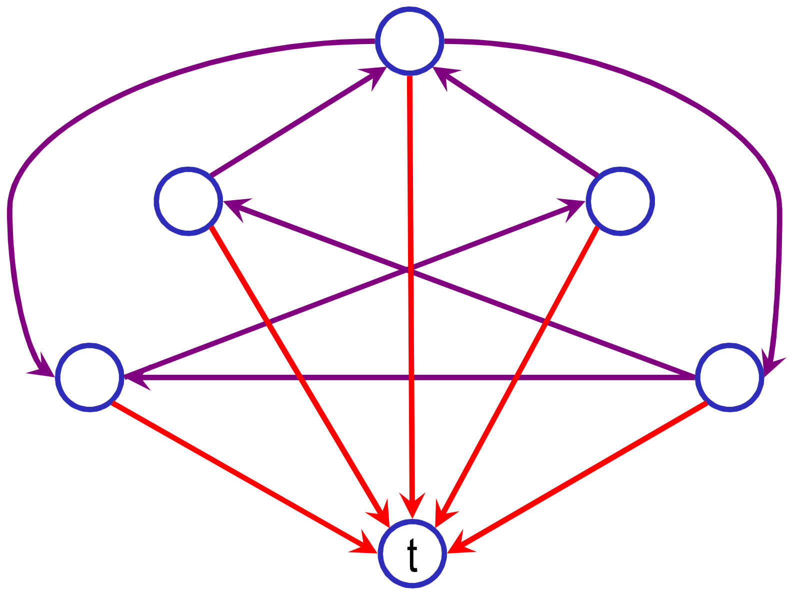


Shortest Path: Finding Negative Cycles

Proof

Let G be a graph

The augmented graph, A(G), is obtained by adding a new node and connecting every node in G with the new node



As is easily seen, G contains a negative cycle if and only if A(G) contains a negative cycle C such that t is reachable from C

QED

Shortest Path: Finding Negative Cycles (cntd)

Extend OPT(i,v) to $i \geq n$

If the graph G does not contain negative cycles then
OPT(i,v) = OPT(n – 1,v) for all nodes v and all $i \geq n$

Indeed, it follows from the observation that every shortest path contains at most n – 1 arcs.

Lemma

There is no negative cycle with a path to t if and only if
OPT(n,v) = OPT(n – 1,v)

Proof

If there is no negative cycle, then OPT(n,v) = OPT(n – 1,v) for all nodes v by the observation above

Shortest Path: Finding Negative Cycles (cntd)

Proof (cntd)

Suppose OPT(n,v) = OPT(n – 1,v) for all nodes v.

Therefore

$$\begin{aligned} OPT(n,v) &= \min\{ OPT(n-1,v),\ \min_{w \in V}\{ OPT(n-1,w) + len(vw) \}\} \\ &= \min\{ OPT(n,v),\ \min_{w \in V}\{ OPT(n,w) + len(vw) \}\} \\ &= OPT(n+1,v) \\ &= \ldots \end{aligned}$$

However, if a negative cycle from which t is reachable exists, then

$$\lim_{i \to \infty} OPT(i,v) = -\infty$$

Shortest Path: Finding Negative Cycles (cntd)

Let v be a node such that $OPT(n,v) \neq OPT(n-1,v)$.

A path P from v to t of weight $OPT(n,v)$ must use exactly n arcs

Any simple path can have at most $n - 1$ arcs, therefore P contains a cycle C

Lemma

If G has n nodes and $OPT(n,v) \neq OPT(n-1,v)$, then a path P of weight $OPT(n,v)$ contains a cycle C, and C is negative.

Proof

Every path from v to t using less than n arcs has greater weight.

Let w be a node that occurs in P more than once.

Let C be the cycle between the two occurrences of w

Deleting C we get a shorter path of greater weight, thus C is negative

The Sequence Alignment Problem

Question:

How similar two words are?

Say "ocurrance" and "occurrence"

They are similar, because one can be turned into another by few changes

gap

oc-urrance

occurrence

mismatch

Clearly, this can be done in many ways, say

oc-urr- ance

occurre-nce

Problem: Minimize the "number" of gaps and mismatches

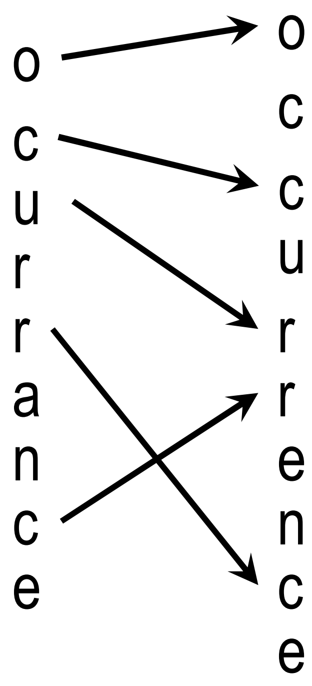
Alignments

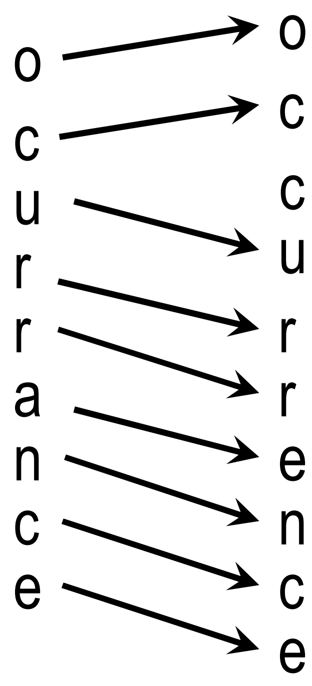
Let $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_n$ be two strings

A matching is a set of ordered pairs, such that an element of each set occurs at most once.

A matching is an alignment if there no crossing pairs:

if (i,j) and (i',j') are in the matching and i < i' then j < j'





The Problem

Let M be an alignment between X and Y.

Each position of X or Y that is not matched in M is called a gap.

Each pair (i,j) ∈ M such that $x_i \neq y_j$ is called a mismatch

The cost of M is given as follows:

- There is δ > 0, a gap penalty. For each gap in M we incur a cost of δ
- For each pair of letters p,q in the alphabet, there is a mismatch cost α_{pq}. For each (i,j) ∈ M we pay the mismatch cost $\alpha_{x_i y_j}$. Usually, $\alpha_{pp} = 0$.
- The cost of M is the sum its gap penalties and mismatch costs

The Problem (cntd)

The Sequence Alignment Problem

Instance:

Sequences X and Y

Objective:

Find an alignment between X and Y of minimal cost.

Dynamic Programming Approach

Lemma

Let M be any alignment of X and Y. If $(m,n) \notin M$, then either the m-th position of X or the n-th position of Y is not matched in M.

Proof

Suppose that $(m,n) \notin M$, and there are numbers $i < m$ and $j < n$ such that $(m,j), (i,n) \in M$.

However, this is a crossing pair.

QED

The Idea

Corollary

In an optimal alignment M, at least one of the following is true

(i) $(m,n) \in$ M; or

(ii) the m-th position of X is not matched; or

(iii) the n-th position of Y is not matched.

Let OPT(i,j) denote the minimum cost of an alignment between

$x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots, y_j$

To get OPT(m,n) we

(i) pay $\alpha_{x_m y_n}$ and then align $x_1, x_2, \ldots, x_{m-1}$ and $y_1, y_2, \ldots, y_{n-1}$ as well as possible, to get

$$OPT(m,n) = OPT(m-1, n-1) + \alpha_{x_m y_n}$$

The Idea (cntd)

(ii) pay a gap cost of δ since the m-th position of X is not matched, and then align $x_1, x_2, \dots, x_{m-1}$ and $y_1, y_2, \dots, y_n$ as well as possible, to get $OPT(m,n) = OPT(m-1,n) + \delta$

(iii) pay a gap cost to get $OPT(m,n) = OPT(m,n-1) + \delta$

Lemma.

The minimum alignment cost satisfy the following recurrence

$$OPT(i,j) = \min\{OPT(i-1,j-1) + \alpha_{x_i y_j}, OPT(i-1,j) + \delta,$$
$$OPT(i,j-1) + \delta\}$$

Moreover, (i,j) is in an optimal assignment for this subproblem if and only if the minimum is achieved by the first of these values.

Alignment: Algorithm

```
Alignment(X,Y)
array  M[0..m,0..n]
set M[i,0]:=iδ for each i
set M[0,j]:=jδ for each j
for i=1 to m do
   for j=1 to n do
    set M[i,j]:=min{M[i-1,j-1]+ α_{x_i y_j}, M[i-1,j]+δ,
                    M[i,j-1]+δ}
   endfor
endfor
return M[m,n]
```

Analysis

Theorem

The Alignment algorithm correctly finds a minimal alignment in $O(mn)$ time

Proof

Soundness follows from previous arguments.

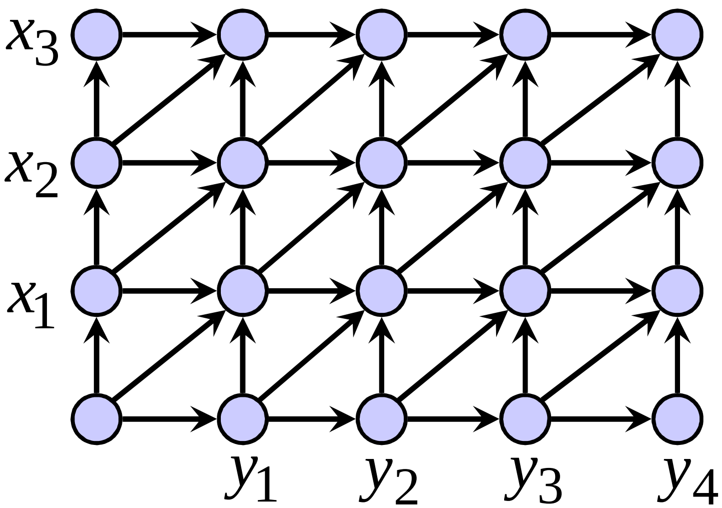
Running time:

We fill up a $m \times n$ table and spend constant time on each entry

QED

Graph Based Approach

Having $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_n$ construct a square grid-like graph



G_{XY}

Weights:

δ on each horizontal or vertical arc

$\alpha_{x_i y_j}$ on the diagonal arc from (i,j) to (i + 1, j + 1)

Lemma

Let f(i,j) denote the minimum weight of a path from (0,0) to (i,j) in G_{XY}. Then for all i,j, we have f(i,j) = OPT(i,j)

Graph Based Approach (cntd)

Proof

Induction on i + j.

Base Case. If i + j = 0, then f(0,0) = 0 = OPT(0,0)

Induction Step.

Suppose the statement is true for all pairs (i', j') with i' + j' < i + j

The last edge on the shortest path to (i,j) is from either (i – 1, j – 1), or (i – 1, j), or (i, j – 1).

Therefore

$$\begin{aligned} & f(i,j) \\ & = \min\{\alpha_{x_i y_j} + f(i-1, j-1), \delta + f(i-1, j), \delta + f(i, j-1)\} \\ & = \min\{\alpha_{x_i y_j} + OPT(i-1, j-1), \delta + OPT(i-1, j), \delta + OPT(i, j-1)\} \\ & = OPT(i,j) \end{aligned}$$

Sequence Alignment in Linear Space

The Alignment algorithm uses O(mn) space, which may be too much

Using an idea similar to that for the Shortest Path problem we can reduce space to linear

We store only two columns of the table

Array B[0..m,0..1] will be used for this purpose

Space Saving Alignment: Algorithm

Space-Saving-Alignment(X,Y)

```
array  B[0..m,0..1]
set B[i,0]:=iδ for each i  /*like column 0 of M
for j=1 to n do
   set B[0,1]:=jδ       /*like M[0,j]
   for i=1 to m do
    set B[i,1]:=min{B[i-1,0]+ α_{x_i y_j}, B[i-1,1]+δ,
                    B[i,0]+δ}
   endfor
   set B[0..m,0]:=B[0..m,1]
endfor
```

Sequence Alignment in Linear Space (cntd)

The Space-Saving-Alignment algorithm runs in $O(mn)$ time and uses $O(m)$ space

Clearly, when the algorithm terminates $B[m,n]$ contains the weight of the optimal alignment

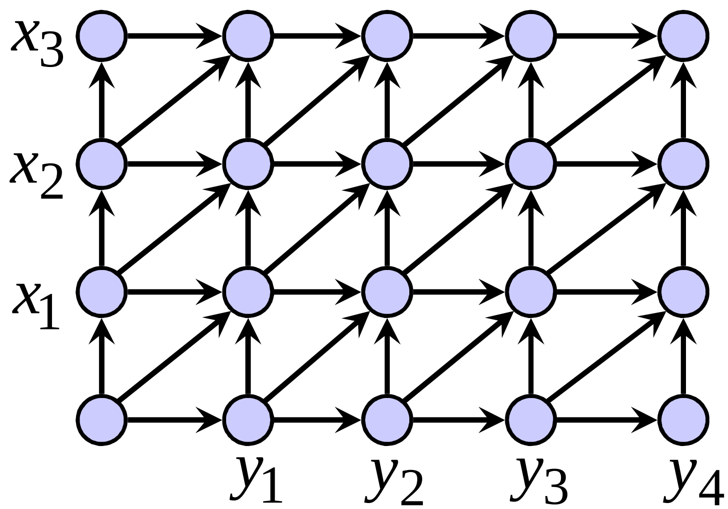
But where is the alignment?

Somehow to find the alignment is more difficult than in the Shortest Path problem

Backward Search

We introduce another function related to OPT

Let g(i,j) denote the length of a shortest path from (i,j) to (m,n)



Lemma

Then for all i,j, we have

$$g(i,j) = \min\{\alpha_{x_{i+1}y_{j+1}} + g(i+1,j+1), \delta + g(i+1,j), \delta + g(i,j+1)\}$$

Backward Search (cntd)

Lemma

The length of the shortest corner-corner path in G_{XY} that passes through (i,j) is f(i,j) + g(i,j)

Proof

Let k denote the length of a shortest corner-to-corner path that passes through (i,j)

It splits into to parts: from (0,0) to (i,j), and from (i,j) to (m,n)

The length of the first part is $\ge$ f(i,j), the length of the second $\ge$ g(i,j)

Thus, $k \ge f(i,j) + g(i,j)$

Finally, the path consisting of the shortest path from (0,0) to (i,j) (it has length f(i,j)), and the shortest path from (i,j) to (m,n) has length exactly f(i,j) + g(i,j)

Backward Search (cntd)

Lemma

Let j be any number $0 \le j \le n$, and let q be an index that minimizes f(q,k) + g(q,k). Then there is a corner-to-corner path of minimum length that passes through (q,k).

Proof

Let k denote the length of a shortest corner-to-corner path in G_{XY}

Fix $j \in \{0, \ldots, n\}$.

The shortest path must use some node in the j-th column. Suppose it is (p,j)

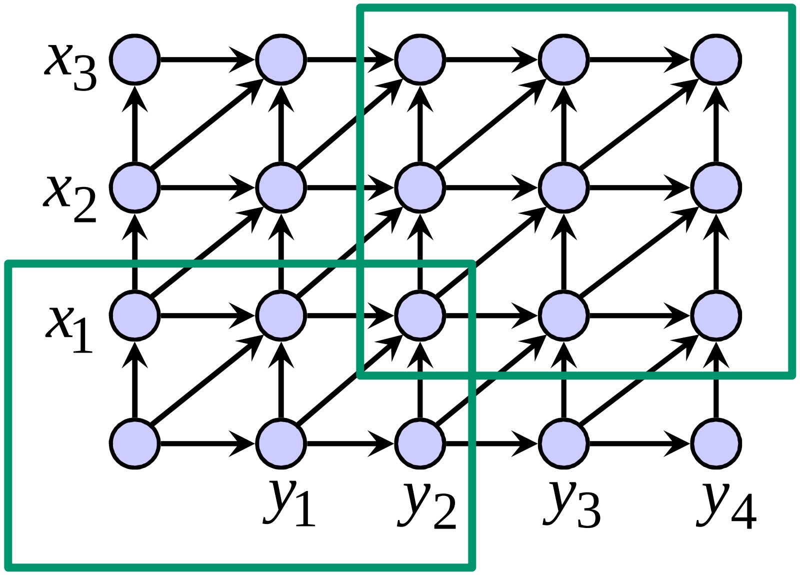
Therefore $k = f(p,j) + g(p,j) \ge \min_q \{ f(q,j) + g(q,j) \}$

If q is the node achieving the minimum, then $k = f(q,j) + g(q,j)$ and

by the previous Lemma there is a shortest path passing through (q,j)

Divide and Conquer

The idea is to split G_{XY} around the middle column and, using the previous Lemma find a node in this column that belongs to a shortest path



We use:

Alignment(X,Y)

Space-Saving-Alignment(X,Y)

Bckw-Space-Saving-Align(X,Y)

Global set P (for the path)

Divide and Conquer (cntd)

Divide-and-Conquer-Alignment(X,Y)

```
set m:=length(X),  n:=length(Y)
if m≤2 or n≤2 do  Alignment(X,Y)
set OPT:=∞ q:=1
for i=1 to m do
  set a:=Space-Saving-Alignment(X[1..i],Y[1..n/2])
  set b:=Bckw-Space-Saving-Align(X[i..m],Y[n/2+1..n])
  if a+b<OPT then do set OPT:=a+b  set q:=i
endfor
add (q,n/2) to P
Divide-and-Conquer-Alignment(X[1..q],Y[1..n/2]
Divide-and-Conquer-Alignment(X[q..m],Y[n/2+1..n]
```

Analysis

Theorem

The Divide-and-Conquer-Alignment algorithm runs in $O(mn)$ time and uses $O(m + n)$ space

Proof

The space complexity is straightforward

Let $T(m,n)$ denote the running time.

The algorithm spends $O(mn)$ on executing Alignment, Space-Saving-Alignment and Bckw-Space-Saving-Align

Then it runs recursively on strings of length $q, n/2$, and $m – q, n/2$.

Thus

$$T(m,n) \le c\cdot mn + T(q,n/2) + T(m - q,n/2)$$

$$T(m,2) \le c\cdot m,$$

$$T(2,n) \le c\cdot n$$

Analysis (cntd)

Proof (cntd)

For a sanity check, suppose $m = n$

Then $T(n) \le 2\, T(n/2) + cn^2$

By the Master Theorem $T(n) = O(n^2)$. So we expect $T(m,n) = O(mn)$

We prove that $T(m,n) \le k \cdot mn$ for some k.

Choosing $k \ge c$ we have the Basis Case:

$$T(m,2) = cm \le 2km, \quad T(2,n) = 2n \le 2kn$$

Suppose $T(m',n') \le k \cdot m'n'$ for all m',n' such that $m'n' \le mn$

$$\begin{aligned} T(m,n) &\le c \cdot mn + T(q,n/2) + T(m-q, n/2) \\ &\le c \cdot mn + kqn/2 + k(m-q)n/2 \\ &= c \cdot mn + kqn/2 + kmn/2 - kqn/2 = (c + k/2) \cdot mn \end{aligned}$$

Choose $k = 2c$