Rank (A[n], k) time

// return the k-th smallest in A[n]

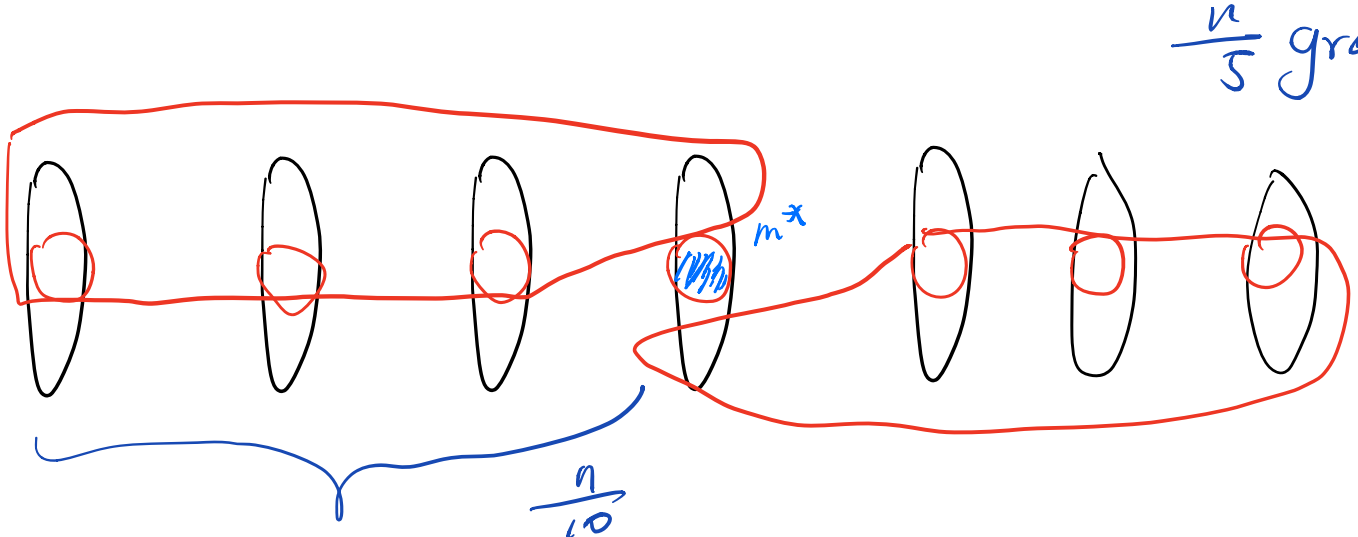
1. if (n ≤ 50)
   brute-force, return O(1)

2. divide A[n] into groups of 5 elements;
   and find the median for each group. O(n)

3. $m^* = \text{Rank}(\text{group-medians}, \frac{n}{10})$ $T(\frac{n}{5})$
   // median of group medians

4. partition A[n] into $A_{\le m^*}$ and $A_{> m^*}$
   → O(n)

5. if ($|A_{\le m^*}| \ge k$)
   return Rank($A_{\le m^*}$, k) $|A_{\le m^*}|, |A_{> m^*}| \le \frac{7n}{10}$
   else return Rank($A_{> m^*}$, $k - |A_{\le m^*}|$) $T(\frac{7n}{10})$



$m^* \geq \frac{3n}{10}$ elements

$\leq \frac{3n}{10}$ elements.

assume the alg takes time $T(n)$

$$T(n) = O(n) + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \leq cn + 10c \cdot \frac{n}{5} + 10c\frac{7n}{10}$$

$$n \to \frac{n}{5} + \frac{7n}{10} = \frac{9n}{10}$$

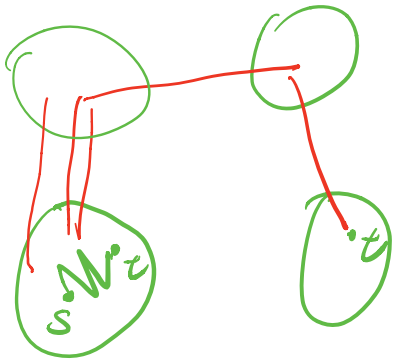
$= 10cn$

$= O(n)$

$T(n) \leq cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$

$T(n) \leq c$ for $n \leq 50$

guess $T(n) \leq 10cn$



Max-bandwidth Path

time

1. remove the smallest half of edges. $O(m)$

2. if [s and t are still in the same piece] $O(m)$
   recursively work on the piece (with $\leq \frac{m}{2}$ edge)

3. else. shrink each piece into a single vertex
   connect the now vertices by smaller edges.
   (if there are parallel, remove all except the one with max bandwidth.) (m)

4. recursively work on the new graph (with $\leq \frac{m}{2}$ edge)

$$T(m) \leq cm + T\left(\frac{m}{2}\right)$$

$$T(m) \leq 2 \cdot c \cdot m$$

Input $(a_1, a_2 \ldots, a_n)$

$\Downarrow$

$((a_1,1), (a_2,2), \ldots, (a_n,n))$

Depth-First-Search

check cycle tree

1. for (each vertex v)
   color[v] = white

1' cc# = 0

2. for (each vertex v)
   if (color[v] == white) DFS(v)

cc# = cc# + 1

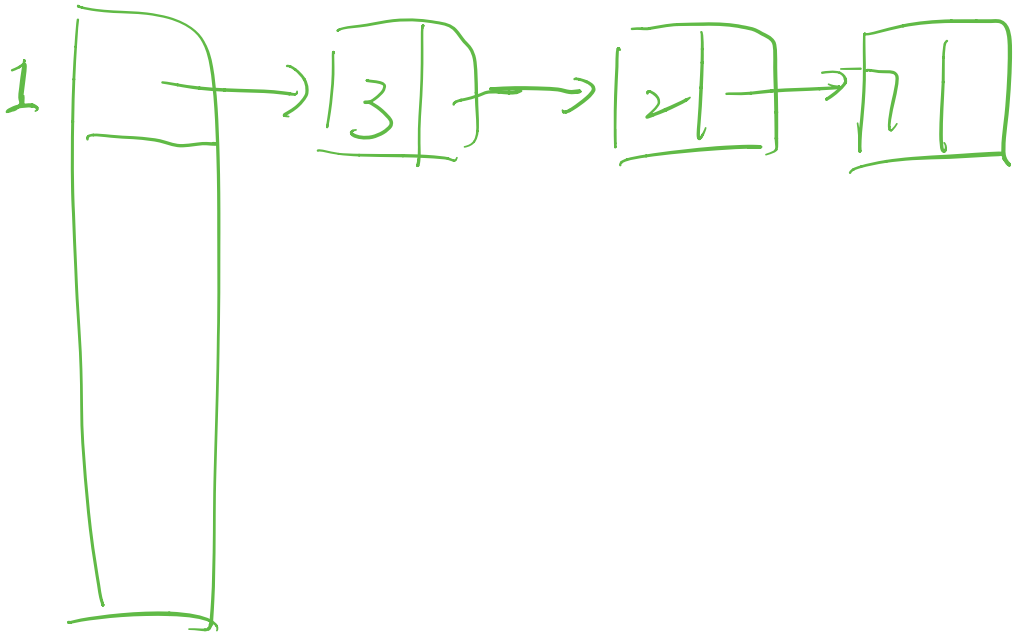
DFS(1)
for each vertex V.
if (color[v] == white)
stop(not connect)
return (connect)

DFS(v)

1. color[v] = gray cc[v] = cc#

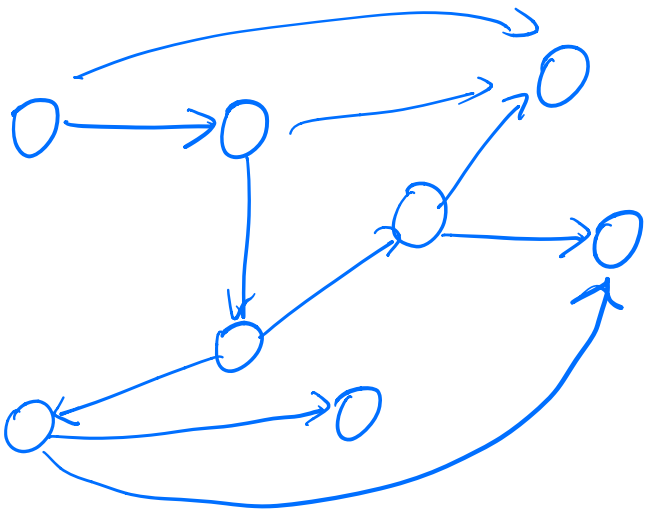
2. for (each edge [v,w])
   if (color[w] == white) DFS(w)

3. color[v] = black.



DFS on directed graphs

input



time = O(m+n)

1. for (each vertex v)
   color[v] = white

1' h = n

2. for (each vertex v)
   if (color[v] == white) DFS(v)

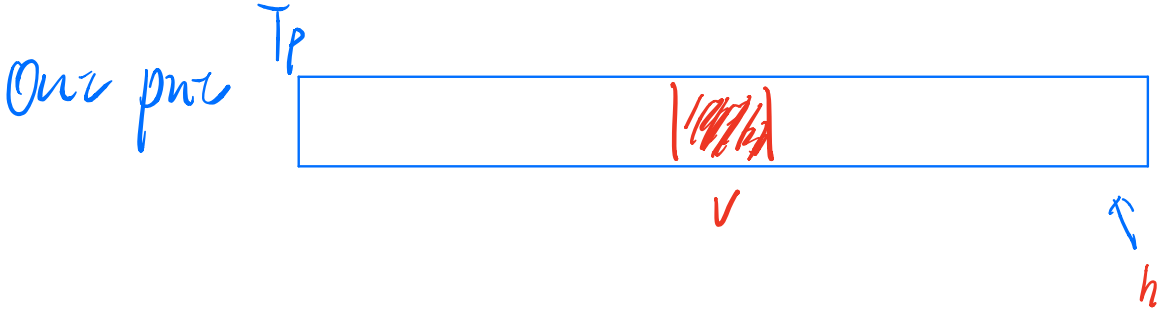
for each vertex, call DFS once.

DFS(v)

1. color[v] = gray

time for all DFS's
$= \sum_{i=1}^{n} deg(v_i)$
$= m$

2. for (each edge [v,w])
   if (color[w] == white) DFS(w)
   else if (color[w] == gray) Stop ('deadlock')
3. color[v] = black; Tp[h--] = v



Claim: if the graph has no cycle, then Tp gives a right order

For each edge $[v, w]$ $\quad v \longrightarrow w$

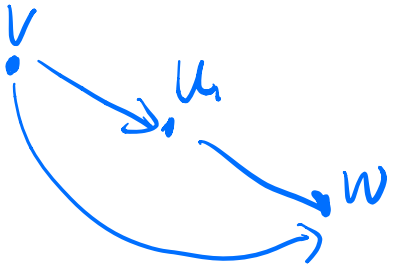
v must appear before w in the array.

Consider DFS(v)

at this point

color[w] = ??

- white
- black
- gray



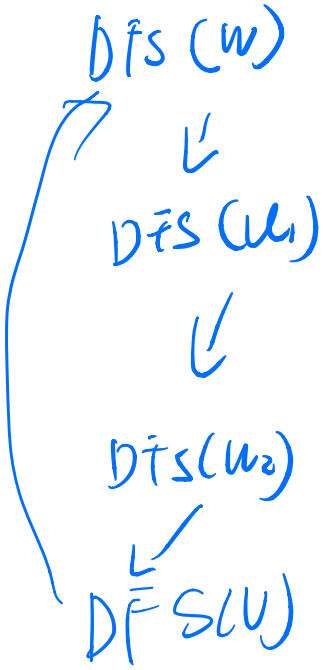
DFS(v)

↓

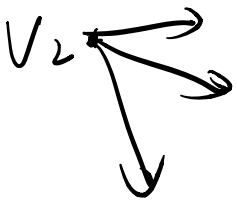
DFS(u)

↓

DFS(w)


Dfs (w)
Dfs (u1)
Dfs(u2)
DFS(v)


V1


V2

1. for (each vertex v)

    color[v] = white

2. for (each vertex v)

    if (color[v] == white) DFS(v)

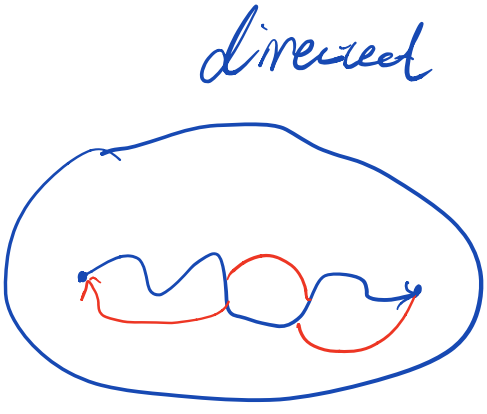
DFS(v)

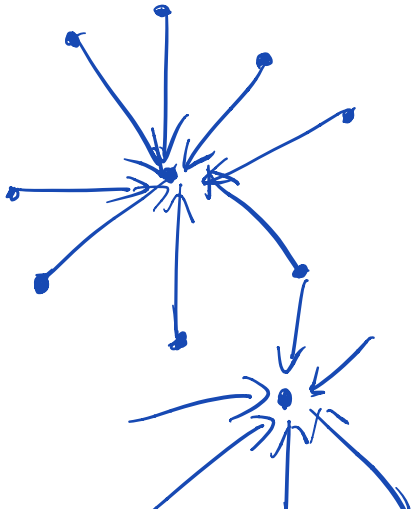
1. color[v] = gray

2. for (each edge [v,w])

    if (color[w] == white) DFS(w)

3. color[v] = black.





a set C of vertices in a directed graph G is a strongly connected component (SCC) if:

1. for each two vertices in C, there are paths going from one to another.

2. C is the maximal

Strongly connected components (SCC)

G: directed graph

a SCC in G is a maximal set of vertices which are mutually reachable

1 no vertex can belong to more than one SCC.

2. every vertex belongs to an SCC

1. for (each vertex v) color[v] = white;

1. h=n;

2. for (each vertex v)
   if (color[v] == white) DFS(v) →

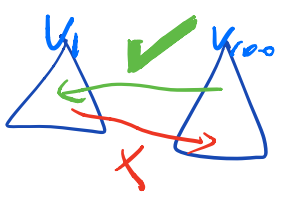
3. let G_R be the reversed G. ☆

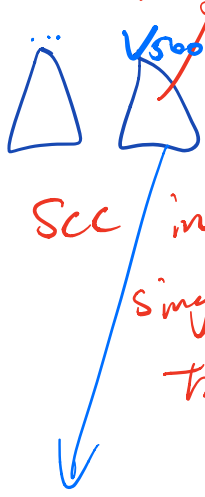
4. for (each vertex v) color[v] = white.

all V_SCC in this tree

4' scc# = 0

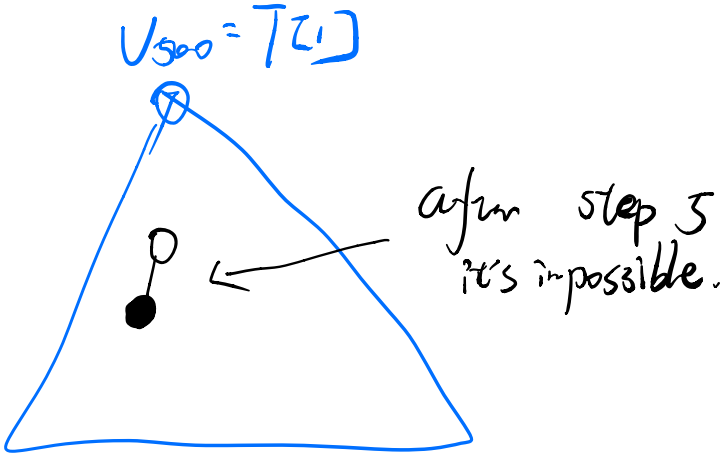
~~5. $DFS_R(T[1])$ // on G_R~~

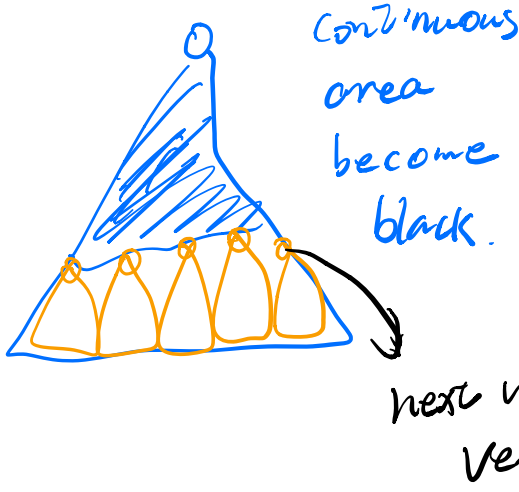


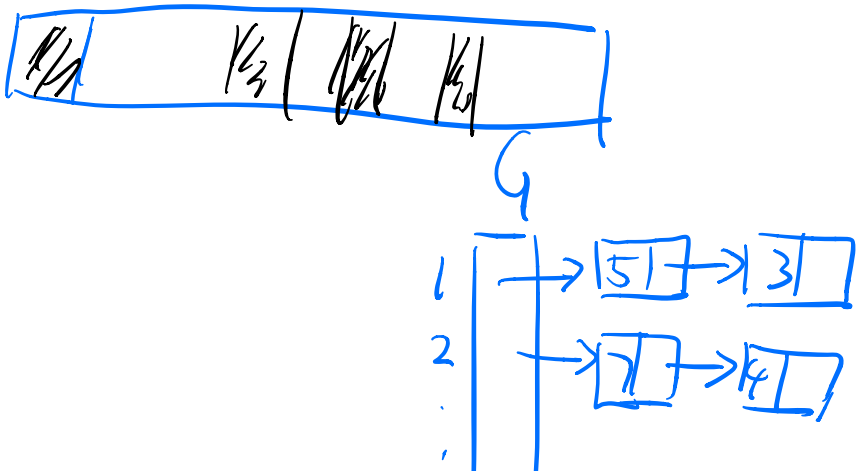


5. for (i=1; i≤n; i++)
   if(color[T[i]] == white
      $DFS_R[T[i]]$
      scc#++

6. output (scc[1,...,n])





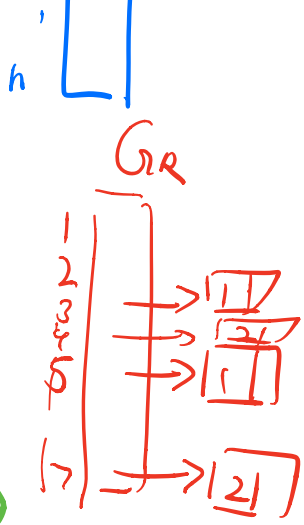


DFS(v)

1. color[v] = gray;

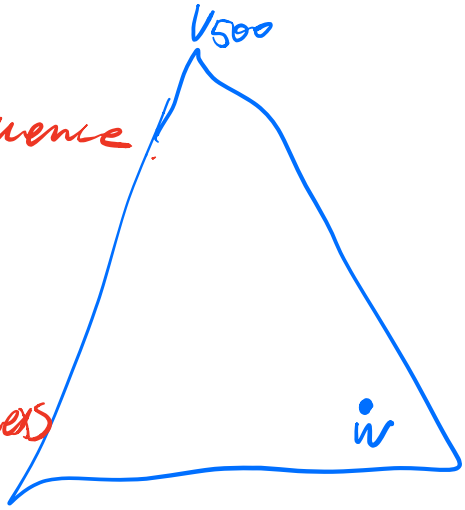
2. for (each edge [v,w])
   if (color[w] == white) DFS(w)

3. color[v]=black ; T[h--]= v.



T 

root of the last tree in sequence



in this tree from which V500 is reachable (they are exactly the desired scc)

DFS$_R$(v)

1. color[v] = gray; scc[v] = scc#

2. for (each edge [v,w])
   if (color[w] == white) DFS$_R$(w);

color[v] = black

A graph is acyclic if it has no cycles

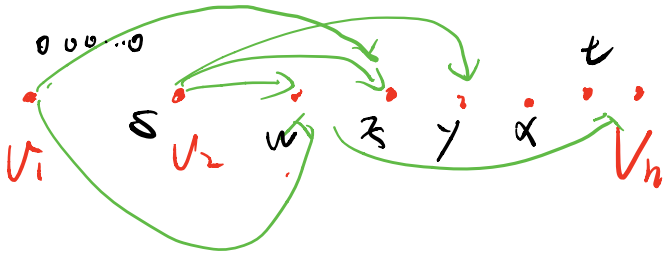
weighted.

Given a directed acyclic graph (DAG) and two vertices s and t in G.

find a longest path from s to t.

1. topologically sort the vertices: $O(m+n)$

$v_1, v_2, \dots, v_n$



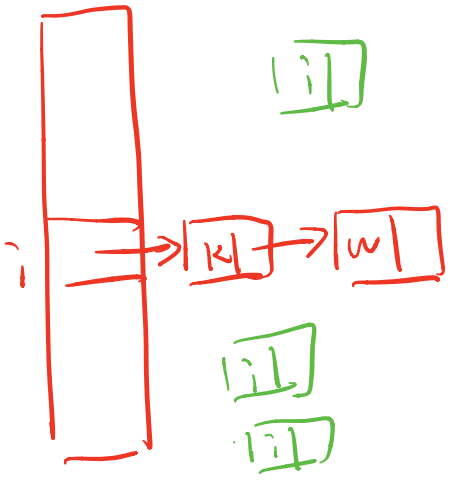
2. for (i=1; i≤n; i++)
   dist[v_i] = $-\infty$

3. Suppose $s = v_k$; dist[s] = 0

4. for (i=k+1; i≤n, i++)
   for(each edge [v_h, v_i])
      if dist[v_i] < dist[v_h] + wt(v_h, v_i)

dist [vi] = dist [vh] + wt (vh, vi);

dad [vi] = vh;



4. for (i=k, i≤n, i++)
    for (each edge [vi, vh])
        if (dist[vh] < dist[vi] + wt(vi, vh))
            dist [vh] = dist [vi] + wt (vt, vh)
            dad [vh] = vi

given 2 sequences

$a_1\ a_2\ a_3\ \cdots\ a_n$

$b_1\ b_2\ b_3\ \cdots\ b_m$

x y x y x y x y x y _

_ y x y x y x y x y x

↙ empty symbol

add "_"s to the sequences to make them of equal length so that the score is maximized.

sequence alignment.

Scores:

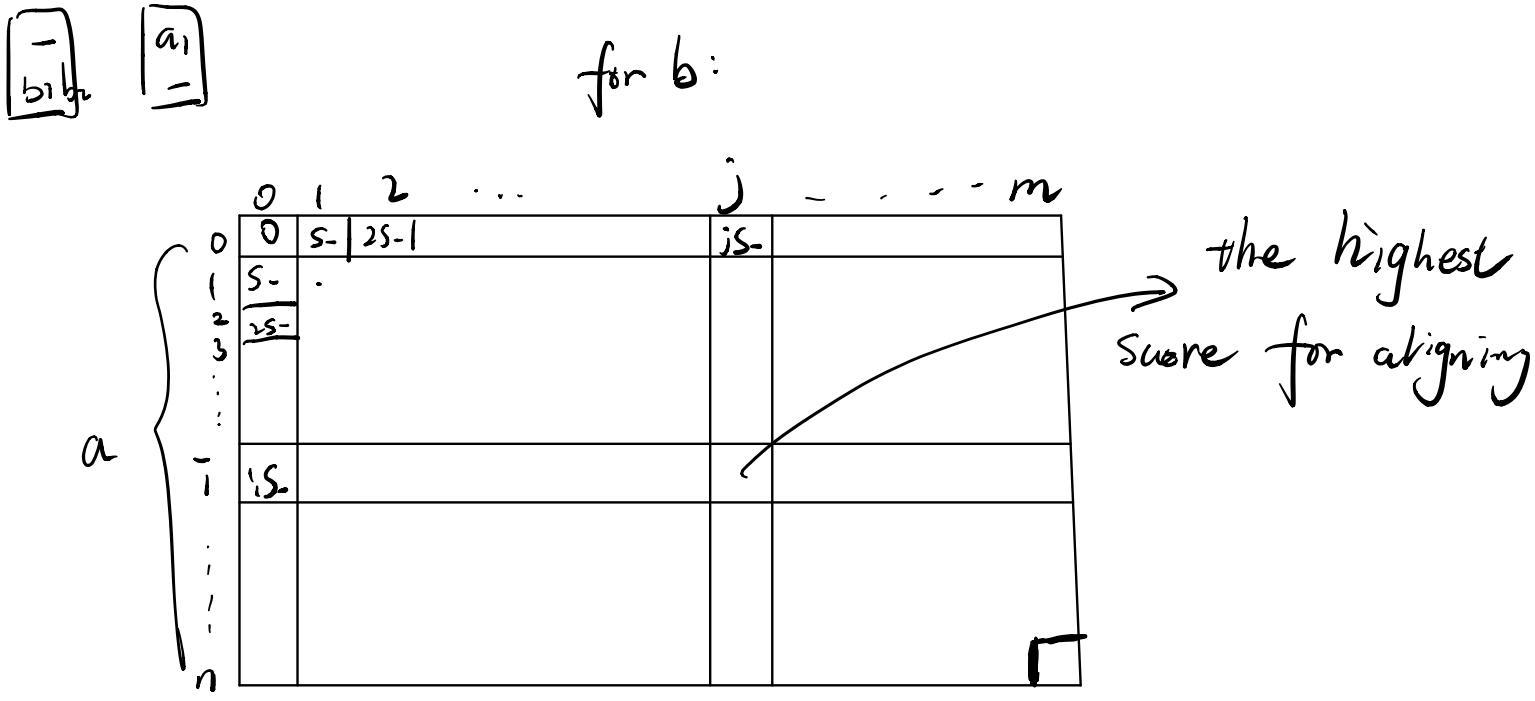
		eg
match:	+	$s_m > 0$
mismatch:	−	$s_{miss} < 0$
"_":	−	$s_- < 0$

$a_1\ a_2\ \cdots\ a_{n-1}\ a_n$

$b_1\ b_2\ \cdots\ \cdots\ b_{m-1}\ b_m$

_

_


-
b1 bj
a1
-
for b:
0 1 2 ... j ... m
0
1
2
3
i
n
a
0
s-
2s-1
js-
s-
2s-
is-
the highest
score for aligning