Dynamic Programming (DP)

All pairs shortest path problem

Used DP : $O(|V|^3)$

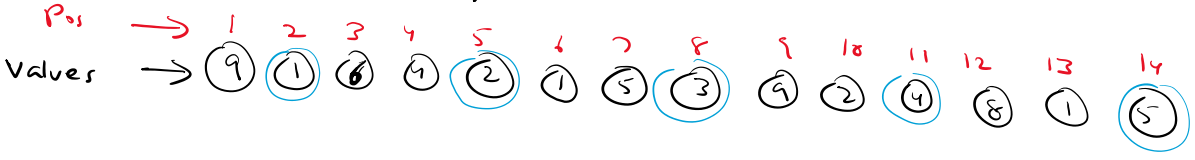
Conceptually: break up the larger problem into smaller "optimal" subproblems

1) recursive definition
2) Initialization (base case)
3) recursive function (bigger in terms of smaller) ← backward
4) forward solution (from smaller to larger)

→ Conceptually: Computational DAG implicit

Longest Increasing subsequence problem (LIS)

Input: sequence of numbers



Subsequence: choose any subset of the positions, but preserve the order

ordered subset of pos <2, 5, 8>

order

Subsequence : (1) (2) (3) ✓

increasing

<3, 6, 9, 12>

(6) (1) (9) (8) ✗

neither increasing, nor decreasing

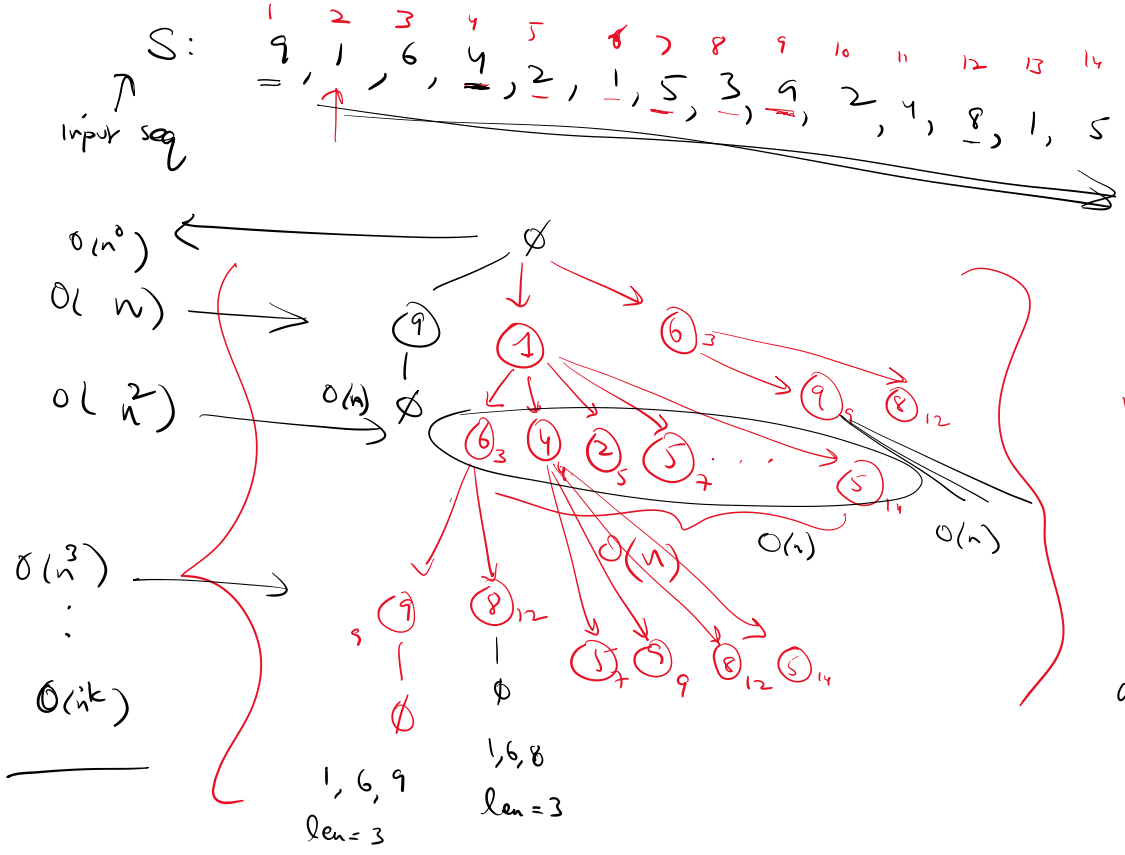
Task: what is the longest increasing subsequence → length (optimization value)
→ actual subsequence (prev)

<2, 5, 8, 4, 14>

$< 2, 5, 8, 4, 14 >$

① ② ③ ④ ⑤ ← output of the algorithm

len = 5



$|S| = n$

# of elements or seq length

Very large search space

What is the size?

Assume that the max length is k

$$\sum_{i=0}^{k} n^i = \frac{n^{k+1} - 1}{n - 1} \approx O(n^k)$$

$n = |S|$

$k =$ max subseq length

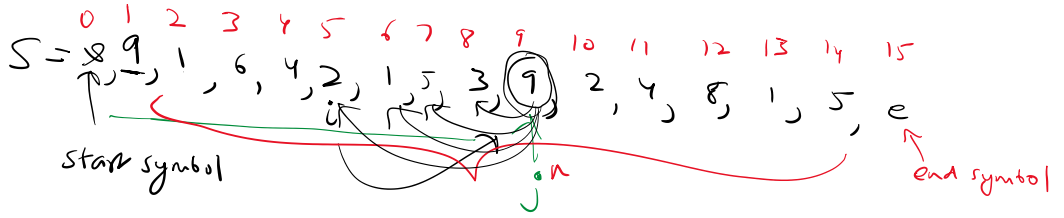
[1 2 3 4 5 6 7 8 9 10 - - - n]

$O(n^n)$ $\quad k = n$

worst case (factorial at least!)

DP for LIS

1) $L[j] \equiv$ longest increasing subsequence ending at pos j in S



2) base case: add a fake value $

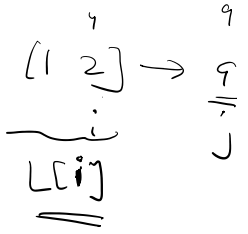
$L[0] = 0$ ← always start at pos 0

$ is smaller than all other values

e is larger than all other value

3) Recursive equation:

$$\{ L[j] = \max_{i<j \text{ (pos)},\ S[i]<S[j] \text{ (value)}} \{ L[i]+1 \} \}$$



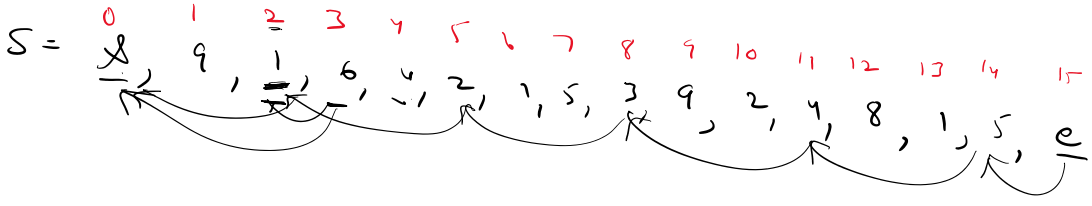
prev[j] = arg max

LIS value: ~~max~~ $\{L[j]\}$ $j=1\ldots n$

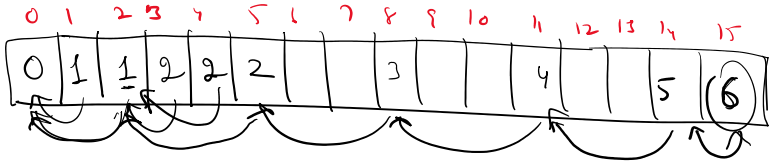
implicit DAG.

$L[n+1] \equiv$ optimal value LIS ← Subtract 1 to get optimal LIS value since we added extra $ and e

4) forward solution:



L =



← implies optimal LIS = 6 − 1 = 5

$L[3]$ = LIS upto pos 3

$|L| = n+2$ ($\frac{n}{\text{LIS}} + 2$)

$O(n)$ elements to compute

at most $O(n)$ steps to compute $L[i]$

$(\frac{n}{2} + 2)$ … $|s|$, s, e

at most $O(n)$ steps to compute $L[j]$

$O(n) \times O(n) = O(n^2)$ indep of $k = |LIS|$

vs

$O(n^k)$ dependent on k

Edit Distance

↳ global alignment

Input: BRILLIANT, THRILLING (BRILLIG)

S_1, S_2

two sequences over some alphabet Σ

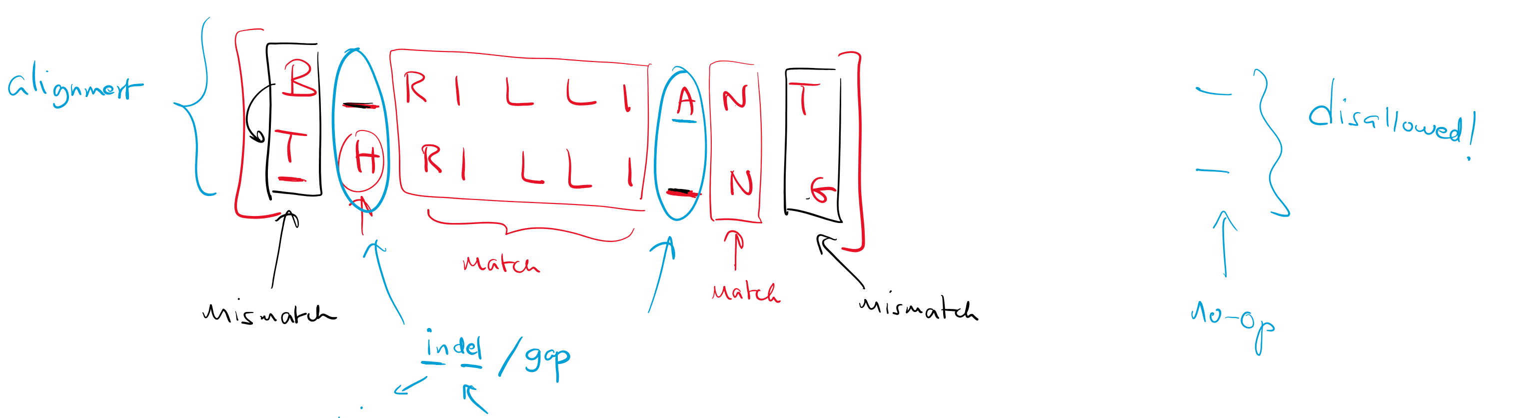
task: find the least # of changes to convert one sequence to the other

edit distance:

1) substitute one letter for another
2) delete / insert one letter

edit cost = 1 – mismatch / gap
0 = match

alignment / correspondance: put the characters in the two sequences in a one-to-one correspondance, allowing for gaps (allow a special character '–')



indel / gap

insert delete

edit distance = #of mismatches + gaps = 2+2 = 4

1 2 3 4 5 6 7 8 9

S_1 = BRILLIANT ⟶ THRILLING = S_2

1. B → T
2. insert H after pos 1 in S_1
3. delete A in S_1
4. T → G

edit distance ≤ 4

Task: find the minimum edit distance between S_1 & S_2

Real-world use case: evolutionary trees (phylogeny)

genes / genome

trivial edit distance = 18

```
[ BRILLIANT_________ ]
[ _________THRILLING ]
```

also a valid alignment!

```
[ BRILLIANT ]
[ THRILLING ]
  ↑ ↑
```

edit = 7

Q: how many alignments are there?

$|S_1| = n$, $|S_2| = m$ $m \leq n$

Ans: Choose $\binom{n+m}{m}$, e.g. if $m = n$

$\binom{2n}{n} \approx \frac{(2n)^{2n}}{n^n}$ exponential

Choose $\binom{2n}{n} \approx O\left(\frac{2^{2n}}{n}\right)$ exponential # of alignments

$\binom{n+m}{m} = \frac{(n+m)!}{n!\,m!}$ Vs $DP = O(mn)$

if $m = n$

$O(n^2)$ polynomial / quadratic

$S_1 = SOAP$, $S_2 = SOMA$

1) $L(i, j) \equiv$ least edit distance between $S_1[1, 2, \ldots, i]$ and $S_2[1, 2, \ldots, j]$

$1 \le i \le n$

$1 \le j \le m$

$|S_1| = n$

$|S_2| = m$

final answer = $L(n, m)$

S_1 _SOAP

S_2 _SOMA

2) base case

$L(0,0) = 0$

$L(0, j) = j \quad \forall\ 1 \le j \le m$

$L(i, 0) = i \quad \forall\ 1 \le i \le n$

3) Recursive equation

$L(i, j) = \ldots \begin{cases} L(i-1, j-1) + \text{edit cost}(S_1[i], S_2[j]) \end{cases}$

$$L(i,j) = \min \begin{cases} L(i-1, j-1) + \text{edit cost}\left(S_1[i], S_2[j]\right) \\ L(i-1, j) + 1 \\ L(i, j-1) + 1 \end{cases}$$

edit cost: 1 mismatch, 0 match

gaps

$$\text{Prev}(i,j) = \arg\min$$

$S_1 = SOAP$

$S_2 = SOMA$

3 Case to consider