1 Mar 2021 — RNA Structure Prediction (§6.5)

Announcements.

1. Problem Set 1 is now graded.
2. Recitations: Some TA's will be providing optional recitations solving sample problems. Stay tuned for announcement on Ed and recitation calendar on website.

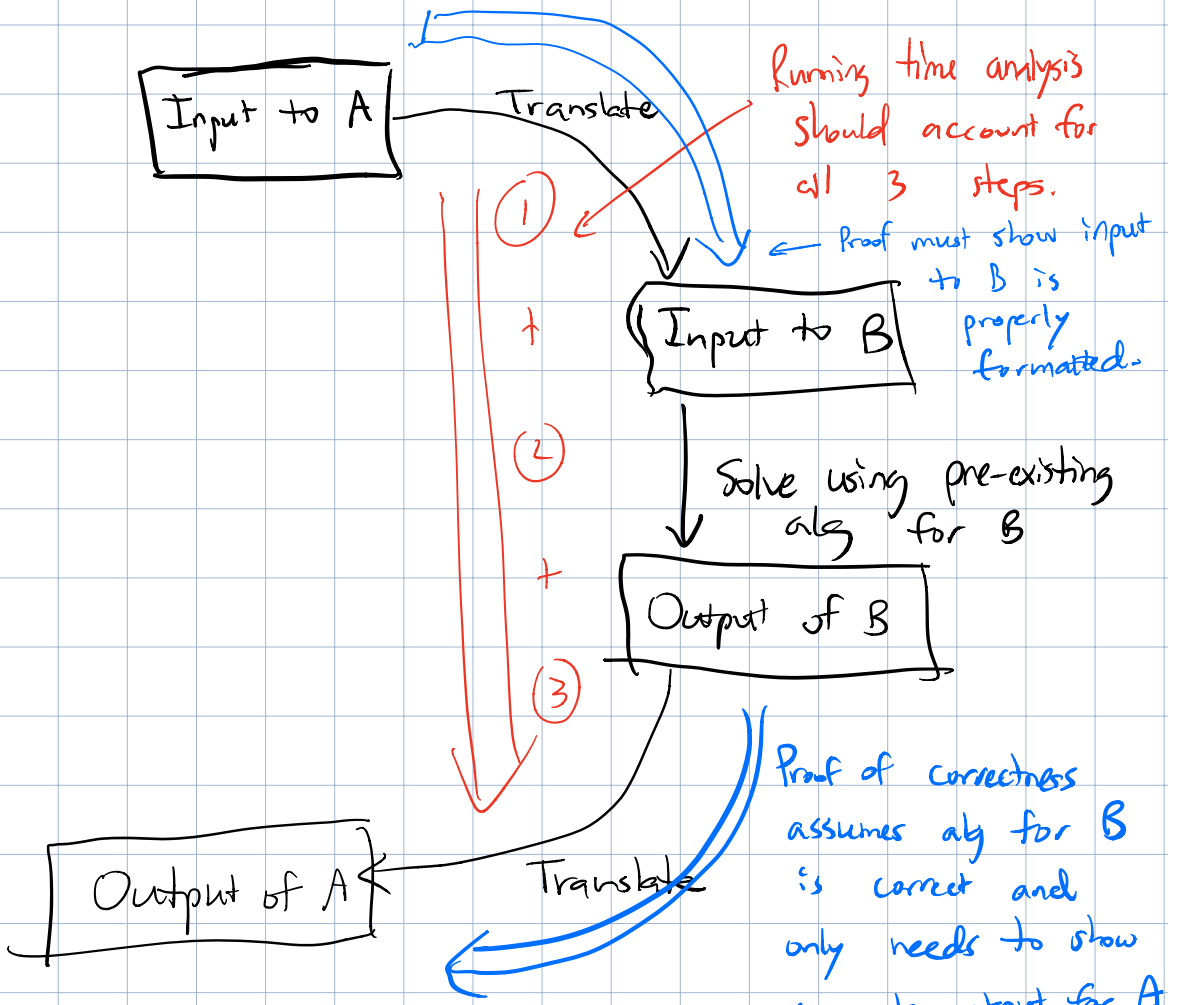
RECAP. Bellman-Ford algorithm takes
- directed graph $G = (V, E)$ with n vertices, m edges
- Source $s \in V$, destination $t \in V$
- cost function $\text{Cost} : E \to \mathbb{R}$ (including possibly negative costs)

and outputs the minimum cost walk from s to t, if one exists, is $O(mn)$ time.

if G has no negative cost cycles that can occur in a walk from s to t

Reducing Problem A to Problem B.

- Using an alg that solves B as a subroutine to solve A.
- Most common type calls to subroutine only once. "Karp reduction"

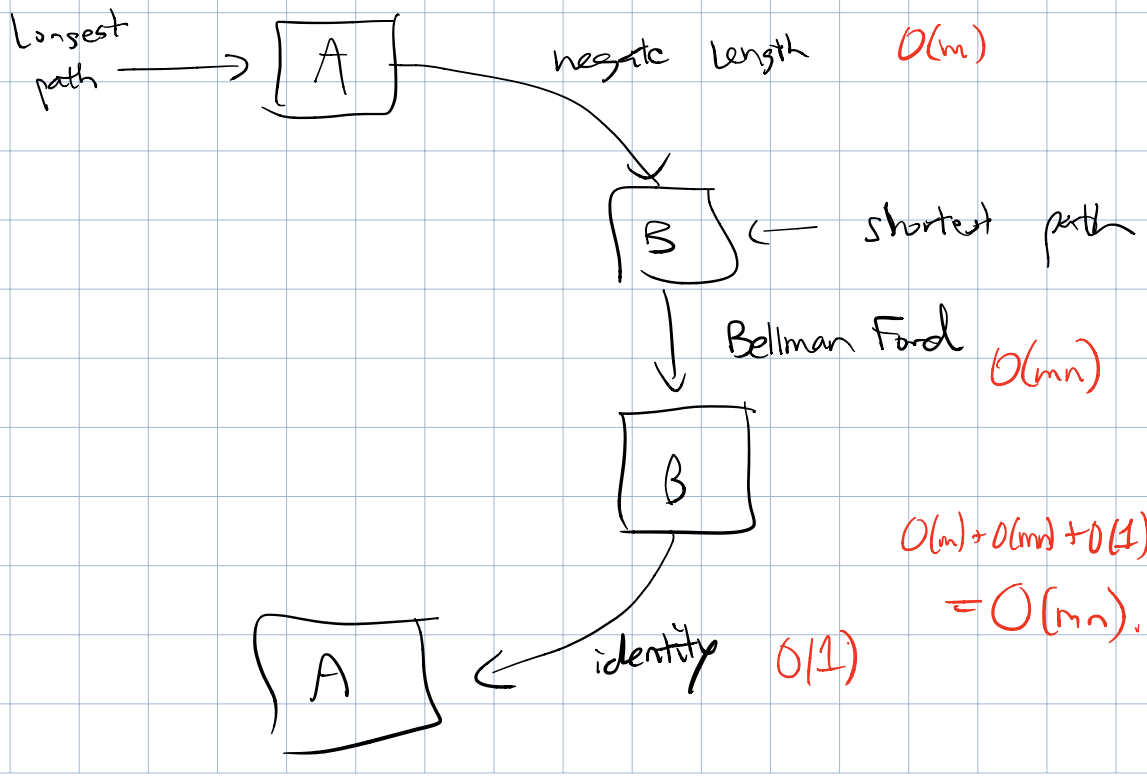


Illustrations...

1. Longest path in a directed acyclic graph.

Given directed $G = (V, E)$ with no cycles.
source s, destination t.
length: $E \to \mathbb{R}$
Find path from s to t with maximum length.

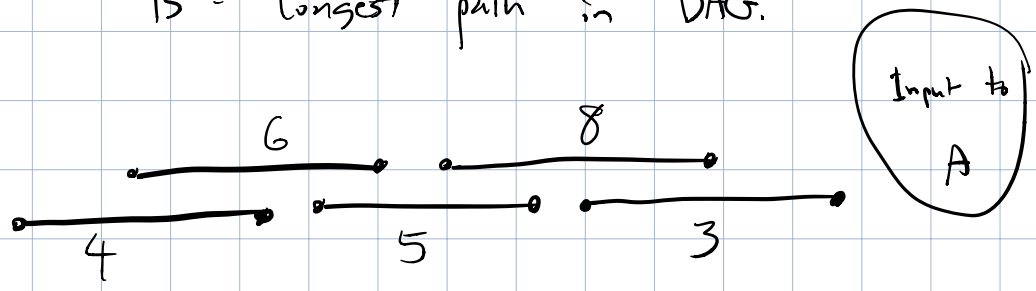
Solution: set $cost(e) = -length(e)$ and run Bellman-Ford.

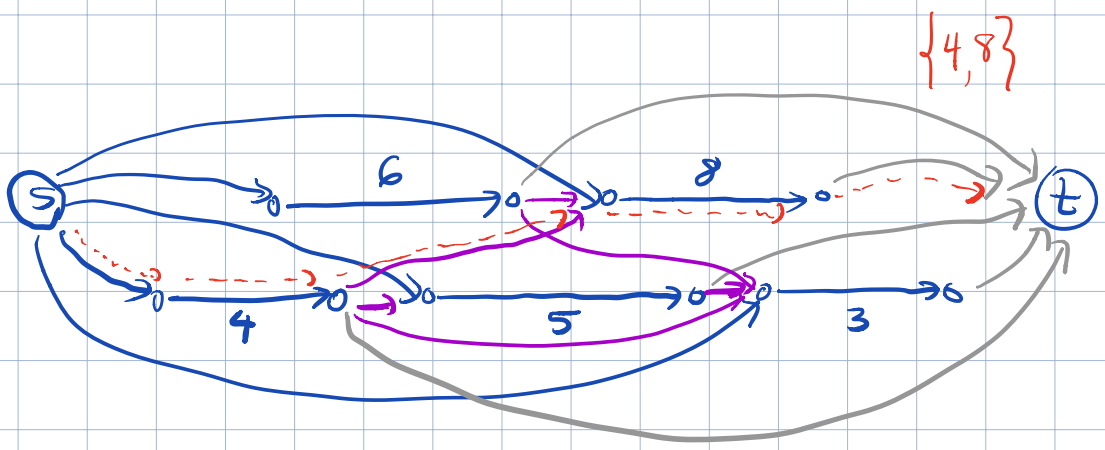


2. Weighted interval scheduling.

A = weight in schd

B = longest path in DAG.





Aside from the 5 edges labeled with numbers, all others have length 0.

This works because

$$\{s\text{-}t \text{ paths in } G\} \longleftrightarrow \{\text{conflict-free sets of intervals}\}$$

via a one-to-one correspondence that transforms path length to weight of an interval set.

Running time: graph has $2n+2$ vertices

Reduction creates $3n$ edges plus any additional "purple" edges between compatible pairs, at most $\binom{n}{2}$ such pairs.

DAG has $N = 2n+2$ vertices

$M = O(n^2)$ edges

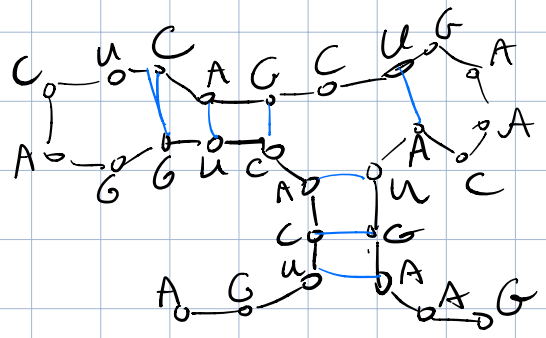
Running time $O(MN) = O(n^3)$.

↑ Remember to translate running time analysis in terms of size of input to B back into an expression that relates it to the input size of problem A.

Exercise: Try translating Friday's knapsack algorithm into a reduction from Knapsack to shortest path (or longest path in a DAG).

RNA secondary structure prediction (§6.5)

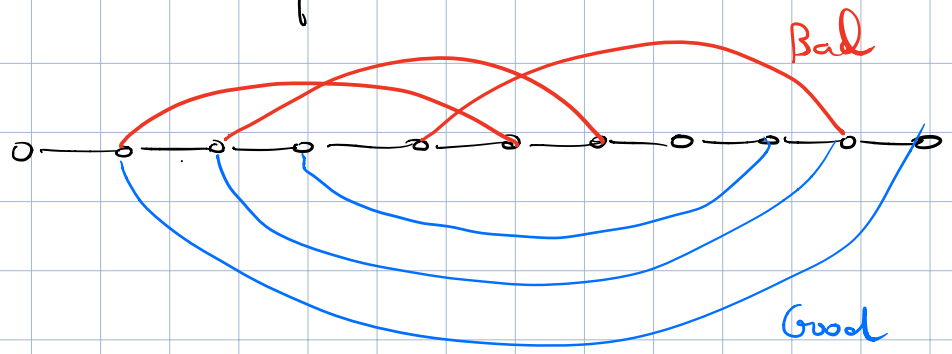
RNA is a single-stranded molecule made of a sequence of "bases" in the set $\{A, C, G, U\}$. These can pair with one another. A pairs with U, C pairs with G... causing the molecule to fold on itself!



A simplified form of the rules for this self-pairing:

1. No sharp turns: if base i pairs with base j then $|i-j| > 4$.
2. A pairs with U, C with G.
3. Matching property: each base pairs with at most one other.
4. Non-crossing: if $i < j < k < \ell$ then

(i,k) and (j,l) cannot both be paired.

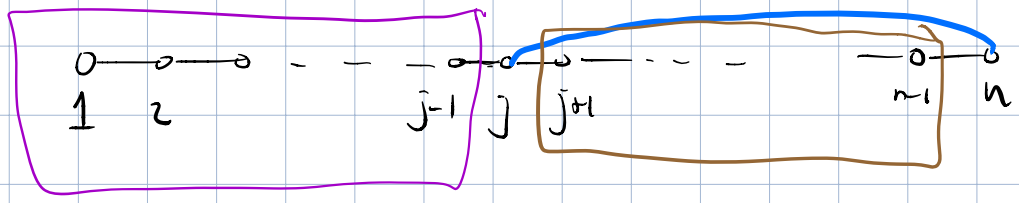


Subject to these rules, the molecule tends to form as many pairs as possible.

The last base in the sequence (base n) is either paired with $j < n-4$ or unpaired.

(a) If unpaired, the best we can do is compute optimal pairing of bases $1, \ldots, n-1$.

(b) If paired with $j \ldots$



Non-crossing $\Rightarrow$ all remaining pairs are contained within the purple segment of the brown segment.

(*) $$OPT(a,b) = \max \left\{ \begin{array}{l} OPT(a, b-1) \\ \max_{j \in P(b)} \{ 1 + OPT(a, j-1) + OPT(j+1, b-1) \} \end{array} \right\}$$

$$P(n) = \left\{ j \;\middle|\; \begin{array}{l} j < b-4 \text{ and} \\ \text{if } b = 'A' \;\; j = 'U' \\ \text{if } b = 'C' \;\; j = 'G' \\ \text{etc} \end{array} \right\}$$

```
for ℓ = 0, ..., n-1
    for a = 1, ..., n-ℓ
        if ℓ = 0
            OPT[a, a+ℓ] = 0
        else
            Compute OPT[a, a+ℓ] using ⊛
        endif
    endfor
endfor
output OPT(1, n).
```

Storing an array element, not calling a recursive function.