CS 401

Dynamic Programming:

RNA Secondary Structure / Negative Shortest Path

Xiaorui Sun

Stuff

Homework 4 has been released last Thursday (due April 19)

- Programming homework on Leetcode
- Submit your code to gradescope
- The first 4 questions are for all the students
- Question 5 is for graduate student only (Undergraduate students who work on Question 5 receive at most 5 bonus points)

RNA Secondary Structure

RNA Secondary Structure (Formal)

Secondary structure. A set of pairs $S = \{(b_i, b_j)\}$ that satisfy:

[Watson-Crick.]

- S is a *matching* and
- each pair in S is a Watson-Crick pair: $A - U$, $U - A$, $C - G$, or $G - C$.

[No sharp turns.]: The ends of each pair are separated by at least 4 intervening bases. If $(b_i, b_j) \in S$, then $i < j - 4$.

[Non-crossing.] If (b_i, b_j) and (b_k, b_l) are two pairs in S, then we cannot have $i < k < j < l$.

Free energy: Usual hypothesis is that an RNA molecule will maximize total free energy.

approximate by number of base pairs

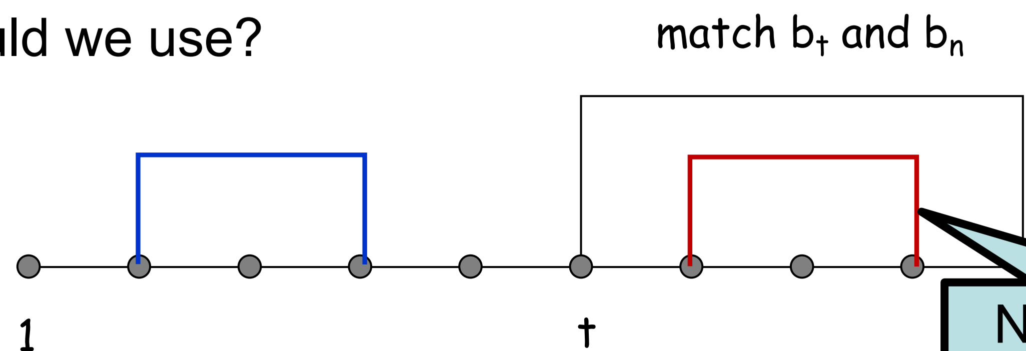
Goal: Given an RNA molecule $B = b_1 b_2 \ldots b_n$, find a secondary structure S that maximizes the number of base pairs.

DP: First Attempt

First attempt. Let $OPT(n)$ = maximum number of base pairs in a secondary structure of the substring $b_1 b_2 \ldots b_n$.

Suppose b_n is matched with b_t in $OPT(n)$.

What IH should we use?



Not correspond to any subproblem

Optimal substructure not exist

Difficulty: This naturally reduces to two subproblems

- Finding secondary structure in $b_1, \ldots, b_{t-1}$, i.e., OP
- Finding secondary structure in $b_{t+1}, \ldots, b_{n-1}$, ???

DP: Second Attempt

Definition: $OPT(i, j)$ = maximum number of base pairs in a secondary structure of the substring $b_i, b_{i+1}, \ldots, b_j$

Case 1: If $j - i \leq 4$.

- $OPT(i, j) = 0$ by no-sharp turns condition.

Case 2: Base b_j is not involved in a pair.

- $OPT(i, j) = OPT(i, j - 1)$

Case 3: Base b_j pairs with b_t for some $i \leq t < j - 4$

- non-crossing constraint decouples resulting sub-problems
- $OPT(i, j) = \max_{i \leq t < j-4} \{ 1 + OPT(i, t - 1) + OPT(t + 1, j - 1) \}$

Recursive Code

```
Let M[i,j]=empty for all i,j.

Compute-OPT(i,j){
   if (j-i <= 4)
     return 0;
   if (M[i,j] is empty)
      M[i,j]=Compute-OPT(i,j-1)
      for t=i to j-5 do
         if (b_t,b_j is in {A-U, U-A, C-G, G-C})
           M[i,j]=max(M[i,j], 1+Compute-OPT(i,t-1) +
                      Compute-OPT(t+1,j-1))
   return M[j]
}
```

Does this code terminate?
What are we inducting on?
Key question: is there any loop in the recursion?

Formal Induction

Let $OPT(i, j)$ = maximum number of base pairs in a secondary structure of the substring $b_i, b_{i+1}, \ldots, b_j$

Base Case: $OPT(i, j) = 0$ for all i, j where $|j - i| \leq 4$.

IH: For some $\ell \geq 4$, Suppose we have computed $OPT(i, j)$ for all i, j where $|i - j| \leq \ell$.

IS: Goal: We find $OPT(i, j)$ for all i, j where $|i - j| = \ell + 1$. Fix i, j such that $|i - j| = \ell + 1$.

Case 1: Base b_j is not involved in a pair.

- $OPT(i, j) = OPT(i, j - 1)$ [this we know by IH since $|i - (j - 1)| = \ell$]

Case 2: Base b_j pairs with b_t for some $i \leq t < j - 4$

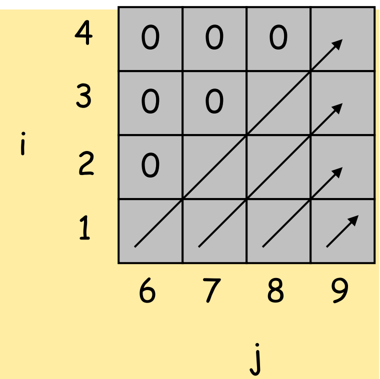
- $OPT(i, j) = \max_{i \leq t < j-4} \{ 1 + OPT(i, t - 1) + OPT(t + 1, j - 1) \}$

We know by IH since difference $\leq \ell$

Bottom-up DP

```
for ℓ = 1, 2, …, n-1
  for i = 1, 2, …, n-1
    j = i + ℓ
    if (ℓ <= 4)
      M[i,j]=0;
      else
        M[i,j]=M[i,j-1]
        for t=i to j-5 do
          if (b_t,b_j is in {A-U, U-A, C-G, G-C})
            M[i,j]=max(M[i,j], 1+ M[i,t-1] + M[t+1,j-1])

  return M[1, n]
}
```



Running Time: $O(n^3)$

Lesson

We may not always induct on i or w to get to smaller subproblems.

We may have to induct on $|i - j|$ or $i + j$ when we are dealing with more complex problems.

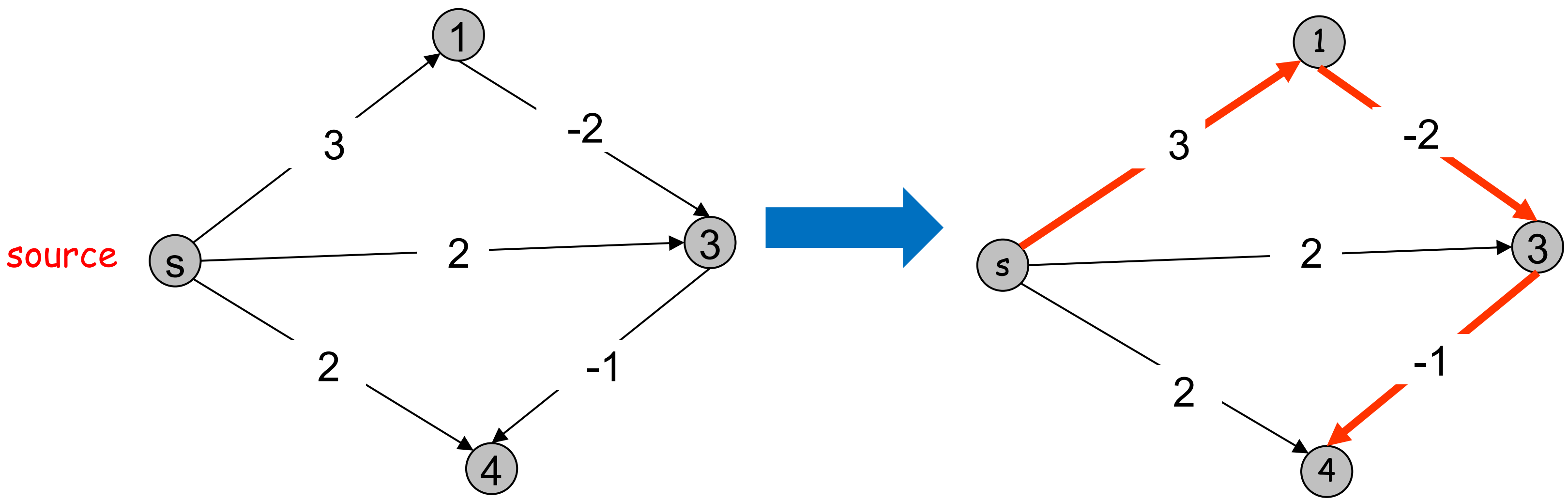
Shortest Paths with Negative Edge Weights

Shortest Paths with Neg Edge Weights

Given a weighted directed graph $G = (V, E)$ and a source vertex s, where the weight of edge (u,v) is $c_{u,v}$ **(that can be negative)**

Goal: Find the shortest path from s to all vertices of G.

Recall that Dikjstra's Algorithm fails when weights are negative

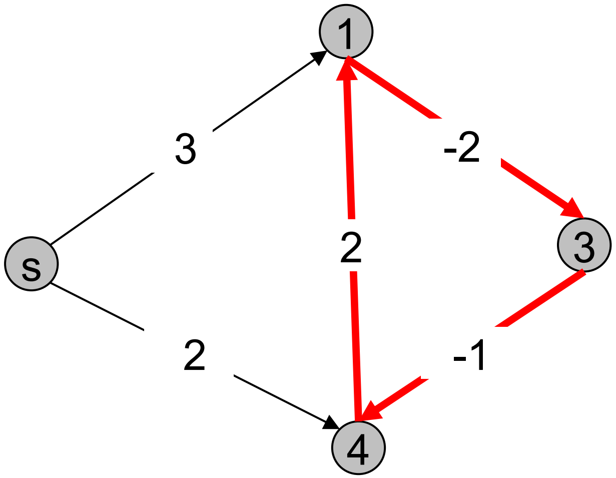


Impossibility on Graphs with Neg Cycles

Observation: No solution exists if G has a negative cycle.

This is because we can minimize the length by going over the cycle again and again.

So, suppose G does not have a negative cycle.



DP for Shortest Path (First Attempt)

Def: Let $OPT(v)$ be the length of the shortest s - v path

$$OPT(v) = \begin{cases} 0 & \text{if } v = s \\ \min_{u:(u,v) \text{ an edge}} OPT(u) + c_{u,v} & \end{cases}$$

The formula is correct. But it is not clear how to compute it.

DP for Shortest Path

Def: Let $OPT(v, i)$ be the length of the shortest s - v path with at most i edges.

Let us characterize $OPT(v, i)$.

Case 1: $OPT(v, i)$ path has less than i edges.

- Then, $OPT(v, i) = OPT(v, i - 1)$.

Case 2: $OPT(v, i)$ path has exactly i edges.

- Let $s, v_1, v_2, \ldots, v_{i-1}, v$ be the $OPT(v, i)$ path with i edges.
- Then, $s, v_1, \ldots, v_{i-1}$ must be the shortest s - v_{i-1} path with at most $i - 1$ edges. So,
$$OPT(v, i) = OPT(v_{i-1}, i - 1) + c_{v_{i-1}, v}$$

DP for Shortest Path

Def: Let $OPT(v,i)$ be the length of the shortest s - v path with at most i edges.

$$OPT(v,i) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{if } v \neq s, i = 0 \\ \min\left(OPT(v,i-1), \min_{u:(u,v) \text{ an edge}} OPT(u,i-1) + c_{u,v}\right) \end{cases}$$

So, for every v, $OPT(v,?)$ is the shortest path from s to v.

But how long do we have to run?

Since G has no negative cycle, it has at most $n-1$ edges. So, $OPT(v,n-1)$ is the answer.

Bellman Ford Algorithm

```
for v=1 to n
    if v ≠ s then
        M[v,0]=∞
M[s,0]=0.

for i=1 to n-1
    for v=1 to n
        M[v,i]=M[v,i-1
        for every edge
            M[v,i]=min(
```

	Complexity	Author
	$O(n^4)$	Shimbel (1955) [30]
	$O(Wn^2m)$	Ford (1956) [14]
*	$O(nm)$	Bellman (1958) [1], Moore (1959) [25]
	$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983) [9]
	$O(\sqrt{n}m \log(nW))$	Gabow and Tarjan (1989) [10]
*	$O(\sqrt{n}m \log(W))$	Goldberg (1993) [12]
*	$\tilde{O}(Wn^\omega)$	Sankowski (2005) [27] Yuster and Zwick (2005) [35]
*	$\tilde{O}(m^{10/7} \log W)$	Cohen, Madry, Sankowski, Vladu (2016)

Table 1: The complexity results for the SSSP problem with negative weights (* indicates asymptotically the best bound for some range of parameters).

$m^{1+o(1)} \log W$ algorithm
By Bernstein, Nanongkai, and Wulff-Nilsen;
Chen, Kyng, Liu, Peng, Gutenberg, Sachdeva
2022

Running Time: $O(nm)$
Can we test if G has negative cycles?
Yes, run for i=1…3n and see if the M[v,n-1] is different from M[v,3n]

DP Techniques

Principle:

- Optimal substructure: Remove certain part of the optimal solution (for the entire problem) is an optimal solution of a subproblem
- Carefully define subproblems. Typically, only a polynomial number of subproblems
- Parameterization/Memorization

Recipe:

- Find optimal substructure by investigating the optimal solution
- Find out additional variables/subproblems that you need to do the induction
- Strengthen the hypothesis and define new subproblems

Dynamic programming techniques.

- Adding a new variable: knapsack.
- Order subproblems in the right way: RNA secondary structure