THE UNIVERSITY OF TEXAS
AT AUSTIN

CS331 ALGORITHM

Assignment 03

Edited by LaTeX

Department of Computer Science

STUDENT

Jimmy Lin

xl5224

INSTRUCTOR

Greg Plexton

TASSISTANT

Chunzhi Zhu

RELEASE DATE

February. 19 2014

DUE DATE

February. 26 2014

TIME SPENT

10 hours

February 20, 2014

Contents

1 Exercise 3

Show how to compute $w(M_i)$ for all i such that $0 \leq i < |A|$ in a total of $O(n)$ time.

1.1 Overall Idea

Since there are n items, if we want to compute all the $w(M_i)$, the technique of dynamic programming is first one to consider. Specifically, we can compute $w(M_0)$, cache it and then compute $w(M_1)$ based on $w(M_0)$ by constant time $\mathcal{O}(1)$. And then we repeat this process for n times. In this manner, we derive all n $w(M_i)$ in $\mathcal{O}(n)$.
In the case, each $w(M_i)$ represents one matching with one linear bid removed under configuration U''.

1.2 Mathematical Notation

Before introducing procedural description and mathematical proof, we first present a series of mathematical notations as follows:

- $w(M_i)$ is the weight of MWMCM $w(M_i)$.
- a_l slope of one linear bid with index l.
- b_l intercept of one linear bid with index l.
- q_l quality of one item with index l.
- m_l offer of one bid with index l.

Besides, all linear bids are sorted in descending order according to the slope. And there is no need to sort the single-item bids in this scenario.

1.3 Algorithmic Description

Given the sorted array of linear bid in terms of its slope and sorted array of single item bid in terms of its offer price.

- First of all, we compute the $w(M_0)$ in $\mathcal{O}(n)$. Specifically:
  - Match all single-item bids to items according to the provided one-one correspondence
  - Remove all matched item from the sorted array of item
  - Match all linear bids to the remained items by virtue of matching the linear bid with the highest slope to the item with the highest quality
- Then, we sequentially compute $w(M_k)$ based on $w(M_{k-1})$ at $\mathcal{O}(1)$. The formula for incremental computation is given as follows:

$$w(M_k) = w(M_{k-1}) + (a_{k-1}q_{k-1} + b_{k-1}) - (a_k q_k + b_k)$$

Since there are one linear time computation and $k-1$ constant time computation. The time complexity of this method is in linear time $\mathcal{O}(n)$.

1.4 Mathematical Proof

Since the computation of $w(M_0)$ as MWMCM in $mcm(G', U'' - A[0])$ is justified in the last assignment, you can reference my solution in the assignment 2 for the proof of that part. And in the following, what I am doing is to validate the formula for incremental computation.

$$w(M_0) = \Sigma_{n\neq 0}(a_n q_n + b_n) + \Sigma_s m_s$$
$$w(M_1) = \Sigma_{n\neq 1}(a_n q_n + b_n) + \Sigma_s m_s$$
$$\vdots$$
$$w(M_k) = \Sigma_{n\neq k}(a_n q_n + b_n) + \Sigma_s m_s$$

We observe that the incremental formula is

$$w(M_k) = w(M_{k-1}) + (a_{k-1}q_{k-1} + b_{k-1}) - (a_k q_k + b_k)$$

Then, we prove it by mathematical induction.

Base case:
for $k = 0$, obviously it does hold.

$$\begin{aligned} w(M_1) &= \Sigma_{n\neq 1}(a_n q_n + b_n) + \Sigma_s m_s \\ &= \Sigma_{n\neq 1}(a_n q_n + b_n) + \Sigma_s m_s + (a_1 q_1 + b_1) - (a_0 q_0 + b_0) - (a_1 q_1 + b_1) + (a_0 q_0 + b_0) \\ &= \Sigma_n (a_n q_n + b_n) + \Sigma_s m_s - (a_0 q_0 + b_0) - (a_1 q_1 + b_1) + (a_0 q_0 + b_0) \\ &= \Sigma_{n\neq 0}(a_n q_n + b_n) + \Sigma_s m_s + (a_0 q_0 + b_0) - (a_1 q_1 + b_1) \\ &= w(M_0) + (a_0 q_0 + b_0) - (a_1 q_1 + b_1) \end{aligned}$$

Inductive cases:
Assume that the relationship between the case of $k-1$ and k holds, now we prove that it also holds for k and $k+1$. ($k+1$ items in total).
That is to say, given

$$w(M_k) = w(M_{k-1}) + (a_{k-1}q_{k-1} + b_{k-1}) - (a_k q_k + b_k)$$

We need to prove

$$w(M_{k+1}) = w(M_k) + (a_k q_k + b_k) - (a_{k+1}q_{k+1} + b_{k+1})$$

By the definition of $w(M_{k+1})$ and $w(M_k)$, it can be easily derived that the inductive cases hold. The specific proof procedure is similar to the base case.

2 Exercise 4

Show how to compute $w(M'_i)$ for all i such that $0 \leq i < |B|$ in a total of $O(n)$ time.

2.1 Overall Idea

This case is slightly more complicated than the previously introduced scenario. This is because the linear bids are the source of instability. However, we can use the same approach as that of solving Exercise 3. Briefly, the base computation for $w(M'_i)$ remained unvaried, whereas the following incremental computation became more complex.

2.2 Mathematical Notation

Everything used in the solution for exercise 3 is still available in this section except that the single-item array should also be sorted in terms of the quality of item it offers to.
That is, for all single-item bids s

$$\text{single[0].offerto.quality} \geq \text{single[1].offerto.quality} \geq \cdots \geq \text{single[s].offerto.quality}$$

Additionally, we have to conceptualize new concepts as follows:

- a'_s represents the slope of linear bid matched to the item, which is originally matched to single-item bid s.
- b'_s represents the intercept of linear bid matched to the item, is originally matched to single-item bid s.
- q_s represents the quality of item which single-item bid s offers to.

2.3 Algorithmic Description

Given the sorted array of linear bids in terms of its slope and sorted array of single-item bids in terms of the **quality of item it offers to**.

- First of all, we compute the $w(M'_0)$ in $\mathcal{O}(n)$. Specifically:
  - Match all single-item bids within $U'' - u$ to items according to the provided one-one correspondence.
  - Remove all matched items from the sorted array of items.
  - Match all linear bids to the remained items by virtue of matching the linear bid with the highest slope to the item with the highest quality.
- Then, we compute $w(M'_m)$ based on $w(M'_{m-1})$ at $\mathcal{O}(1)$
  - Compute the update on matching made by single-item matching, that is to add the offer price of single-item bid indexed by $s-1$ back and subtract the offer price of single-item bid indexed by s.
  - Compute the update on matching made by linear bid. It is a little more complicated since the change of single-item bid will deprive the original matching made by linear bids. We rematch all the items whose quality is between the quality of the item we added and that of the item we subtracted.
- To summarize, at the step above the mathematical formula for incremental computation is given as follows:

  $$w(M'_m) = w(M'_{m-1}) + U_{single} + U_{linear}$$

  where U_{single} corresponds to update on single-item bid matching,

  $$U_{single} = m_{s-1} - m_s$$

and U_{linear} corresponds to update on linear bid matching,

$$U_{linear} = \Sigma_{q_{s-1} \leq q_j \leq q_s}(q_j * (a'_{j-1} - a'_j))$$

Since there are one linear time computation and $m-1$ constant time computation. Precisely speaking, a particular incremental computation for each $w(M'_m)$ is not constant but all the whole stage of incremental computing is in linear time. Thus, The time complexity of this method is in linear time $\mathcal{O}(n)$.