Tutorial

1. Negative edge weights

Dijkstra's algorithm, unmodified, can't handle some graphs with negative edge weights. Your friend has come up with a modified algorithm for finding shortest paths in a graph with negative edge weights:

1. Find the largest negative edge weight, call this weight $-w$.
2. Add w to the weight of all edges in the graph. Now, all edges have non-negative weights.
3. Run Dijkstra's algorithm on the resulting non-negative-edge-weighted graph.
4. For each path found by Dijkstra's algorithm, compute its true cost by subtracting w from the weight of each of its edges.

NO, the algorithm will not work as he think, since # edges is not confirmed that the decreased component is not calculated as well. you'd better calculate # edges and corresponding component is # scores = w · # edges

Will your friend's algorithm work?

2. Master Theorem

The Master Theorem states that if we have a recurrence relation $T(n)$ such that

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d),$$
$$T(1) = c,$$

then,

$$T(n) \in \begin{cases} \Theta\left(n^d\right) & \text{if } a < b^d \\ \Theta\left(n^d \log n\right) & \text{if } a = b^d \\ \Theta\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}.$$

Note: a similar formula holds for O and Ω as well.

Apply the Master Theorem to find the time complexities of the following recurrence relations (in Big-Theta terms).

a. $T(n) = 9T\left(\frac{n}{3}\right) + n^3,\ T(1) = 1$ $a=9,\ b=3,\ c=1,\ d=3 \Rightarrow a < b^d \Rightarrow T(n) \in \Theta(n^3)$

b. $T(n) = 64T\left(\frac{n}{4}\right) + n + \log n,\ T(1) = 1$ $a=64,\ b=4,\ c=1,\ d=1 \Rightarrow a > b^d \Rightarrow T(n) \in \Theta(n^{\log_b a}) = \Theta(n^3)$

c. $T(n) = 2T\left(\frac{n}{2}\right) + n,\ T(1) = 1$ $a=2,\ b=2,\ c=1,\ d=1 \Rightarrow a = b^d \Rightarrow T(n) \in \Theta(n \log n)$

d. $T(n) = 2T\left(\frac{n}{2}\right),\ T(1) = 1$ $a=2,\ b=2,\ c=1,\ d=0 \Rightarrow a > b^d \Rightarrow T(n) \in \Theta(n^{\log_2 2}) = \Theta(n)$

3. Lower bound for the Closest Pairs problem

The closest pairs problem takes n points in the plane and computes the Euclidean distance between the closest pair of points.

The algorithm provided in lectures to solve the closest pairs problem applies the divide and conquer strategy and has a time complexity of $O(n\log n)$.

The *element distinction problem* takes as input a collection of n elements and determines whether or not all elements are distinct. It has been proved that if we disallow the usage of a hash table[1] then this problem cannot be solved in less than $n\log n$ time (*i.e.*, this class of problems is $\Omega(n\log n)$).

Describe how we could use the closest pair algorithm from class to solve the element distinction problem (where the input is a collection of floating point numbers), and hence explain why this proves that the closest pair problem must not be able to be solved in less that $n\log n$ time (and is thus $\Omega(n\log n)$).

4. Horspool's Algorithm

Use Horspool's algorithm to search for the pattern GORE in the string ALGORITHM.

5. Horspool's Algorithm Continued

How many character comparisons will be made by Horspool's algorithm in searching for each of the following patterns it the binary text of one million zeros?

a. 01001

b. 00010

c. 01111

6. (Homework) Horspool's Worst-Case Time Complextity

Using Horspool's method to search in a text of length n for a pattern of length m, what does a worst-case example look like?

- [1]Excluding the use of a hash table is done to conform to the *algebraic decision tree* model of computation, where we cannot use the values of the elements to index into memory, and only allows us to compute and compare the elements themselves (and simple functions of these elements).↩

Question 3

We can view each element as $(C_i, 0)$, which will make the elements behave like a series of consistent points on the $X - axis$, by using the Closest Pairs problem, we can select the closest points on the line, and what we need to do is just check whether the distance is zero, then we can know whether the element is distinct, and since the ELEMENTDISTINCTION is $\Omega(nlogn)$ so the closest points pair can not run faster than it, we can have the conclusion that the closest points pair is $\Omega(nlogn)$

Question 4

we can find that

S[]	G	O	R	X (other letters)
value	3	2	1	4

- O fails to match with E so the shift is 2 positions
- I fails to match with E so the shitf is 4 positions
- return failure

A	L	G	O	R	I	T	H	M	
G	O	R	E						
		G	O	R	E				
						G	O	R	E

Question 5

1. The pattern's last 1 will be compared against every single 0 in the text (except of course the first four), since the skip will be 1. So 999,996 comparisons.
2. Here we will will make two comparisons between shifts, and each shift is of length 2. So the answer is again 999,996 comparisons
3. For the last pattern, the skip is 4. So we will make 249,999 comparisons.

Question 6

Consider the case that P is 1000000 and the T is 000000000000000 all ones, assume the length of T is n while the length of P is m, each time for the comparation is m, $\#times = n - m + 1$, so the times of comaprations is $\#compare = m \times (n - m + 1) \in O(nm)$