The Chinese University of Hong Kong, Shenzhen

CSC 3100

Data Structures

Programming Assignment 4

Author:
Shi Wenlan

Student Number:
119010265

December 25, 2021

Contents

1 Binary Search Tree

1.1 What are the possible solutions for this problem?

This question consists of two sub-questions:
1. How to judge whether this tree is a binary search tree or not.
2. How to count the leaves of a tree.

In the process of solving the first sub-problem, I thought of a total of two ideas:
1. Add edges according to the input parent-child relationship, discuss and exclude situations that do not belong to the binary search tree.
2. Add an edge based on the input child, and then check whether the actual parent of the child matches the parent in the input, so that several situations can be discussed less.

In the process of solving the second sub-problem, I found that the number of leaves of the tree only increases when the number of childen of a nodes in the tree increses to 2. So I add a attribute *num_child* to the *node* struct to record the number of the child of a node.

1.2 How do you solve this problem?

I tried both of them and adopted idea 2 to solve the first sub-problem in the end. The core idea of the algorithm is shown in the figure:

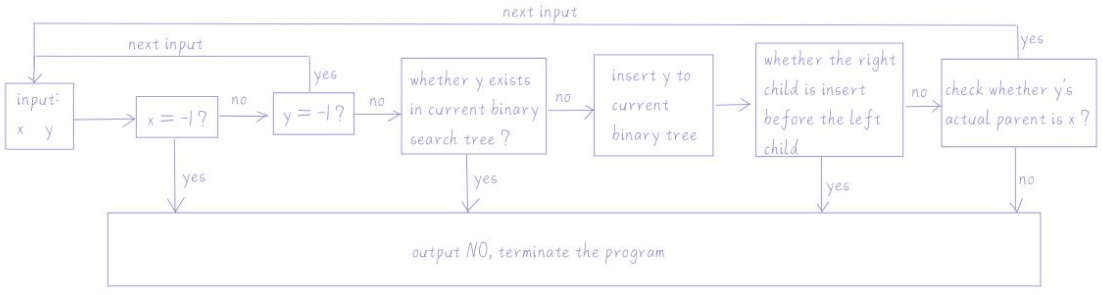

Figure 1: Algorithm Idea

1.3 Why is your solution better than others?

This method can rule out more abnormal situations. A more detailed description is shown in the figure below:

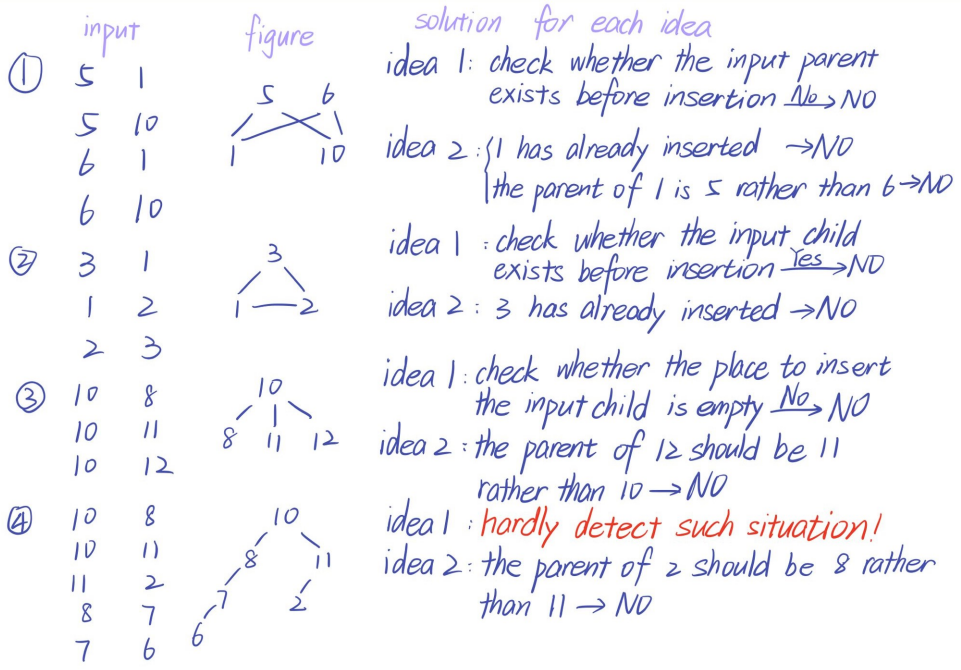

Figure 2: Description of Abnormal Situations

Even though the fourth case does not appear in the test cases on OJ, I still feel it is necessary to take it into consideration.

In addition, this algorithm has also given satisfactory results at a relatively fast speed. The following figure is the output result of the code test on OJ.

测试情况 #1:	AC	0.044s	1.62 MB	(10/10)
测试情况 #2:	AC	0.032s	1.62 MB	(10/10)
测试情况 #3:	AC	0.047s	1.62 MB	(10/10)
测试情况 #4:	AC	0.036s	1.62 MB	(10/10)
测试情况 #5:	AC	0.054s	1.62 MB	(10/10)
测试情况 #6:	AC	0.049s	1.62 MB	(10/10)
测试情况 #7:	AC	0.039s	1.62 MB	(10/10)
测试情况 #8:	AC	0.049s	1.62 MB	(10/10)
测试情况 #9:	AC	0.037s	1.62 MB	(10/10)
测试情况 #10:	AC	0.041s	1.62 MB	(10/10)

Resources: 0.430s, 1.62 MB

Figure 3: Result of BST on OJ

1.4 How do you test your program?

Submit codes to OJ to see the results.

2 Dijkstra

2.1 What are the possible solutions for this problem?

The total idea to solve this problem is as figure 4, I keep on optimize the implementation so that it can pass more tests.

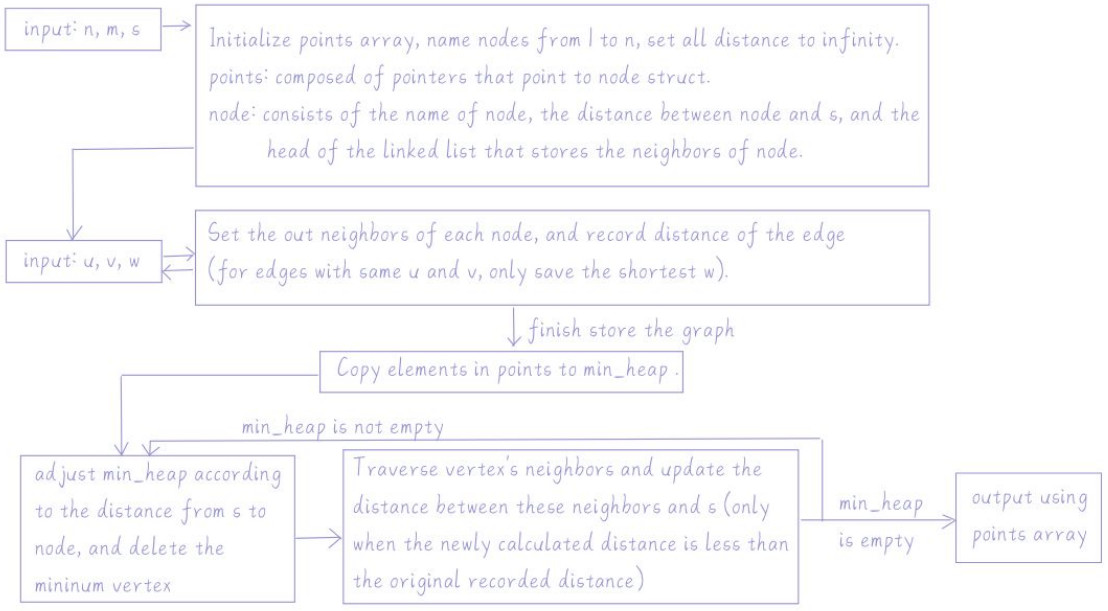

Figure 4: Algorithm Idea

2.2 How do you solve this problem?

In the process of solving this problem, I update the implementation 3 times:

Implementation 1: Just as the idea shown in figure4, subsequent attempts are optimized on its basis. (Note that the min_heap array and the points array store node pointers instead of node structures.)

Implementation 2: Implementation 2 is an optimization of implementation 1, and the main difference lies in the sorting operation of min˙heap. In implementation 1, I sort the entire min˙heap by perlocateDown from V/2 to 1 after finishing the relax operations of all neighbors of the vertex to be visited in the loop(O(N^2)). And in implementation 2, I use a perlocateDown of 1 after deleteMin and a perlocateUp after relax operation(O(ElgN)).

In addition, implementation 2 also optimizes implementation 1 in other details:
1. The name attribute of node is deleted, and the name of node is equal to the position of node in points.
2. Points and min˙heap are initialized at the same time, and both of them discard position 0, and start recording data from position 1.
3. The position attribute of node is added, which is used to record the position of node in min˙heap.

Implementation 3: Implementation 3 is a further optimization based on implementation 2. This optimization mainly optimized the following two points:
1. No longer consider filtering duplicate edges, each edge is directly added to the head of the neighbor linked list.
2. IO optimization: replace the original scanf with the inline function that uses getchar to achieve the same effect. Thereby greatly improving the speed of getting input.

2.3 Why is your solution better than others?

I wrote the codes of all algorithms respectively, and the results of the first implementation is too slow as shown in following figure:

测试情况 #1:	AC	0.026s	1.62 MB	(0/0)
测试情况 #2:	AC	0.035s	1.62 MB	(0/0)
测试情况 #3:	AC	0.029s	2.83 MB	(0/0)
测试情况 #4:	AC	0.184s	6.18 MB	(0/0)
测试情况 #5:	TLE	>1.000s	37.64 MB	(0/10)
测试情况 #6:	TLE	>1.000s	68.06 MB	(0/10)
测试情况 #7:	TLE	>1.000s	68.06 MB	(0/10)
测试情况 #8:	TLE	>1.000s	48.72 MB	(0/10)
测试情况 #9:	TLE	>1.000s	51.30 MB	(0/10)
测试情况 #10:	TLE	>1.000s	78.89 MB	(0/10)
测试情况 #11:	TLE	>1.000s	109.31 MB	(0/10)
测试情况 #12:	AC	0.544s	18.30 MB	(10/10)
测试情况 #13:	AC	0.555s	18.30 MB	(10/10)
测试情况 #14:	TLE	>1.000s	46.14 MB	(0/10)

Figure 5: Result of Implementation 1

The results of the second implementation is faster than implementation 1, but it is still too slow to pass all the tests as shown in following figure:

测试情况 #1:	AC	0.026s	1.62 MB	(0/0)
测试情况 #2:	AC	0.026s	1.62 MB	(0/0)
测试情况 #3:	AC	0.028s	2.83 MB	(0/0)
测试情况 #4:	AC	0.070s	6.18 MB	(0/0)
测试情况 #5:	AC	0.467s	37.64 MB	(10/10)
测试情况 #6:	AC	0.917s	68.32 MB	(10/10)
测试情况 #7:	AC	0.916s	68.32 MB	(10/10)
测试情况 #8:	AC	0.247s	48.72 MB	(10/10)
测试情况 #9:	AC	0.292s	51.71 MB	(10/10)
测试情况 #10:	AC	0.763s	78.89 MB	(10/10)
测试情况 #11:	TLE	>1.000s	109.31 MB	(0/10)
测试情况 #12:	AC	0.219s	18.30 MB	(10/10)
测试情况 #13:	AC	0.219s	18.30 MB	(10/10)
测试情况 #14:	AC	0.641s	46.14 MB	(10/10)

Figure 6: Result of Implementation 2

The results of the third implementation is faster than implementation 2. It passed all the tests at a satisfactory speed:

×14

测试情况 #1:	AC	0.049s	1.62 MB	(0/0)
测试情况 #2:	AC	0.035s	1.62 MB	(0/0)
测试情况 #3:	AC	0.055s	2.68 MB	(0/0)
测试情况 #4:	AC	0.060s	6.03 MB	(0/0)
测试情况 #5:	AC	0.278s	37.48 MB	(10/10)
测试情况 #6:	AC	0.531s	68.28 MB	(10/10)
测试情况 #7:	AC	0.572s	68.28 MB	(10/10)
测试情况 #8:	AC	0.232s	48.57 MB	(10/10)
测试情况 #9:	AC	0.281s	51.15 MB	(10/10)
测试情况 #10:	AC	0.505s	78.73 MB	(10/10)
测试情况 #11:	AC	0.769s	109.16 MB	(10/10)
测试情况 #12:	AC	0.133s	18.15 MB	(10/10)
测试情况 #13:	AC	0.141s	18.15 MB	(10/10)
测试情况 #14:	AC	0.434s	45.99 MB	(10/10)

Figure 7: Result of Implementation 3

2.4 How do you test your program?

Submit codes to OJ to see the results.