note @161 125 views

Notes for Final Exam P4

As many students misunderstood what problem 4 was asking about when examing and confusing about the score, here is a unified explanation. Problem 4 *gives* a sort algorithm which can output the smallest half array, and requires you to prove that this algorithm *at least* make $\Omega(nlogn)$ camparisons. Hint tells you any comparison-based algorithm which sort n numbers need $\Omega(nlogn)$ camparisons.

1. This is a proof problem of lower bound, not design problem.
2. The problem has given an algorithm, though no one knows the concrete content of it. Any proof of a certain algorithm such as mergesort, quicksort is not accepted.
3. Time complexity is not comparison, this problem is not about time complexity.
4. $\Omega(nlogn)$ is not $O(nlogn)$.

A correct proof is use this unknown algorithm to construct a sort algorithm which can sort n numbers and use contradiction to prove lower bound, just like a reduction proof in some ways.

Update: Decision Trees will be regrade as appropriate.

exam

Updated 4 months ago by 邬梦莹

followup discussions *for lingering questions and comments*

[✓ Resolved] Unresolved

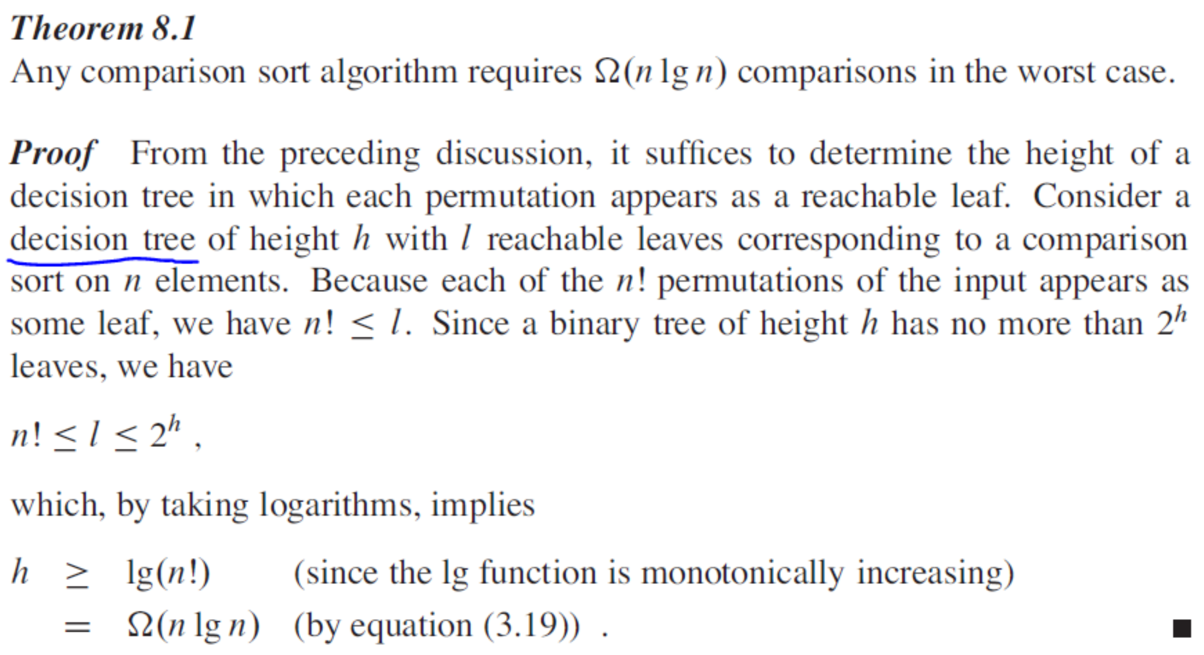
Anonymous Comp 5 months ago
Using contradiction is the only correct answer? That might be too rigid.
helpful! | 0

邬梦莹 5 months ago
Other reasonable proof can also be accepted, but exploration about the inside of the algorithm is unacceptable.

good comment | 0

[✓ Resolved] Unresolved

Anonymous Scale 5 months ago
Does this note mean that we can only use the contradiction method? That might be too rigid.
And there exists a proof (Theorem 8.1) based on **decision tree** in ***Introduction to Algorithms, C. E. Leiserson, C. Stein, T. H. Cormen, and R. Rivest, (third edition) page 193.***


Theorem 8.1
Any comparison sort algorithm requires $\Omega(n \lg n)$ comparisons in the worst case.

Proof From the preceding discussion, it suffices to determine the height of a decision tree in which each permutation appears as a reachable leaf. Consider a decision tree of height h with l reachable leaves corresponding to a comparison sort on n elements. Because each of the $n!$ permutations of the input appears as some leaf, we have $n! \le l$. Since a binary tree of height h has no more than 2^h leaves, we have

$$n! \le l \le 2^h ,$$

which, by taking logarithms, implies

$$\begin{aligned} h &\ge \lg(n!) && \text{(since the lg function is monotonically increasing)} \\ &= \Omega(n \lg n) && \text{(by equation (3.19))} . \end{aligned}$$

■

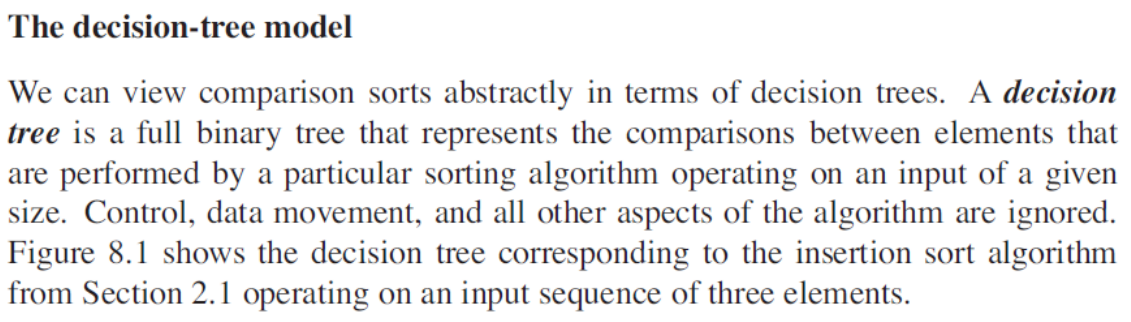
helpful! | 0

邬梦莹 5 months ago
Theorem 8.1 has been given in the hint, there is no difference between using this proof and use the hint to proof directly. The algorithm in the problem should be a black box and the only thing you can use is its input and output. Any exploration of the inside of the box is not accepted.

good comment | 0

Anonymous Scale 5 months ago
In the description of the Q4,“ The algorithm is only allowed to make comparison between a pair of numbers ” , and considering that we need output n/2 smallest numbers , I think all these mean using comparison-based algorithm to sort n numbers then output the smaller half. The problem is as same as the hint's description. Why can't we prove the hint to solve this problem ? Why a general proof cannot be used to solved the specific problem?

And **decision-tree model here is a abstract model for all comparison-based sort algorithm not a certain algorithm** you mentioned in the note above, according to *Introduction to Algorithms, C. E. Leiserson, C. Stein, T. H. Cormen, and R. Rivest, (third edition) page 192.*


The decision-tree model

We can view comparison sorts abstractly in terms of decision trees. A ***decision tree*** is a full binary tree that represents the comparisons between elements that are performed by a particular sorting algorithm operating on an input of a given size. Control, data movement, and all other aspects of the algorithm are ignored. Figure 8.1 shows the decision tree corresponding to the insertion sort algorithm from Section 2.1 operating on an input sequence of three elements.

helpful! | 1

逄 白 5 months ago “I think all these mean using comparison-based algorithm to sort n numbers then output the smaller half.”

This is a faulty intuition: you cannot specify an implementation to prove the lower bound. A counter-example is that you may first find the median, then sort directly on the smaller half, leaving the larger half unsorted.
helpful! | 0

余春霖 5 months ago 1) No matter what algorithm is used to sort, as long as it is comparison-based, we can use decision tree to represent this algorithm.

2) At every leaf of decision tree is a sorting result of the smallest n/2 elements. Since there are C(n, n/2)*(n/2)! different results, and the number of leaves must be larger than the number of different results, then we can*

obtain the number of leaves is larger than C(n, n/2)(n/2)! =n!

3) we can obtain the height of the binary tree >= log(n!) , which is Ω(nlogn)

actually I don't think comparison-based algorithms have any difference in nature, if my way of proving the problem is wrong, then why will the proof of the theorem/hint still hold?
helpful! | 1

逢 白 5 months ago “The problem is as same as the hint's description.”

This is correct since you can give a *very* simple reduction: assume you want to sort (3,1,2), just input (3,1,2,4,4,4).
helpful! | 0

逢 白 4 months ago What I can do is to point out inaccuracy that might have led to falty proof since I don't know your exact answer.
1) I agree. You can definitely prove by decision-tree, as long as you're giving proof to the "half sorting" problem, not to the "total sorting" one. So you can't just prove the hint, which is a conclusion for the latter;
2) & 3) I think that proof is acceptable if you provided it during exam.

The slight difference between two problems cannot be ignored when rigour matters. You may call it "trivial relationship", but never "the same".
helpful! | 0

余春霖 4 months ago thanks for ur reply :). Above is similar to I wrote in exam and I think there is no difference of logic for proving the hint and the problem since their number of possible orders is bounded by n!.
helpful! | 0

Rui Fan 4 months ago Your answer is basically correct, but you should explain it in more detail to receive full credit. In particular, you should say something like "There are n! possible inputs (ie orderings of n numbers). The correct output (ie the smallest n/2 numbers in sorted order) is an ordered subset of size n/2 out of the n inputs, ie we need to choose n/2 locations out of n locations, and the ordering of the values at those locations matters. So there are >= m = C(n, n/2) (n/2)! = n(n-1)...(n/2+1) outputs, so the decision tree has height >= log m = Omega(n log n)."

Also, your calculation is incorrect, because C(n,n/2)(n/2)! = n!/(n/2)! = n(n-1)...(n/2+1).
good comment | 1