5. Lower bound

2018年10月10日 15:02

Definitions

Worst Case Complexity of Problem P

$C_A: \mathbb{N} \to \mathbb{N}$ the WC runtime of alg A of size n

$C(P) = \min\{ C_A \mid A \; solves \; P \}$

Want to get a bigger $C(P)$ which tighter bounds problem P

Worst Case Complexity of Problem P in a class $\mathcal{A}$

$C(P) = \min\{ C_A \mid A \in \mathcal{A} \wedge A \; solves \; P \}$

$\mathcal{A} \subseteq \mathcal{A}' \Rightarrow C(P)$ for $\mathcal{A}'$ is also a $C(P)$ for $\mathcal{A}$

Want to get a more powerful model (larger class)

The class focuses on the class of comparison

Comparison Tree

Each internal node is labelled by a comparison

Each edge out of a internal node is labelled by a different possible outcome of that comparison (typically T/F)

Each leaf is labelled by an output

There is a different tree for each different input size, and WC complexity = height of such tree

Average case complexity = $\sum_{l:leaf} depth(l) \times Prob(input \; leads \; to \; l)$

Information Theory lower bounds

A t-ary (t branches) tree of height h has $\leq t^h$ leaves

A t-ary tree of L leaves has height $\geq ceil(\log_t L)$

Therefore, WC number of t-ary comparisons performed by a comparison tree T that solves $P \geq ceil(\log(\#leavs \; in \; T))$

$C(P) \geq \min\{ ceil(\log_t(\#leavs \; in \; T)) \mid T \; solves \; P \}$

Information Theory gives

P has $\geq m$ different posssible outputs IMPLIES all comparison trees solves P has $\geq m$ leaves, thus has height $\geq ceil(\log_\mathrm{t} m)$

Example 1 P: searching a sorted list using only $\leq$ comparisons

Input: $A[1..n]$ sorted; x: key

Output: i: $A[i] == x$; 0 otherwise

There are $n+1$ possible outputs, hence $C(P) \geq ceil(\log_2(n+1))$

Consider another problem P'

Output: i: $A[i] == x$; $-\infty$: $x < A[1]$; ∞: $x > A[n]$; $(i, i+1)$: $A[i] < x < A[i+1]$

There are $2n+1$ possible outputs, hence $C(P) \geq ceil(\log_2(2n+1))$

Claim Any comparison tree that solves P' can be converted into a comparison tree that solves P by relabelling leaves, and converse is also true (proven in lemma).

Lemma in any comparison tree that solves P using $\leq$ comparisons, for any array A, if search key $y, z, y < z$ goes to the same leaf, then search key u also goes to that leaf for $y < u < z$

Proof Consider any comparison on the path that leads to the output, say the comparison is $x \leq A[j]$.

Suppose the comparison gives T, then $u < z \leq A[j]$, u goes to the same edge as z

Suppose the comparison gives F, then $A[j] > y > u$, u goes to the same edge as y

Therefore, $C(P) \geq ceil(\log_2(2n+1))$

Example 2 U:searching unsorted list using only ==

Input: $A[1..n], x: key$

Output: $i: x == A[i]; 0: x \notin A$

There are $n+1$ possible outputs

$C(U) \geq ceil(\log_2(n+1))$

Example 3.1 Sorting using only $\leq$

Input: $A[1..n]$

Output: a permutation π of $\{1, \dots, n\}$ $s.t. A[\pi(i)] \leq A[\pi(i+1)], i \in \{1, \dots, n-1\}$

There are $n!$ different permutations

Hence $C(S) \geq ceil(\log_2 n!) \in \Theta(n \log n)$

Example 3.2 Consider $countsort(A)$

Input: $A[1..n], A[i] \in \mathbb{N}. A[i] \leq k$

Procedure: creating an empty array C of size k, filled with 0, traverse A and each time $C[A[i]]++$, finally give the number i $C[i]$ times

The runtime is $O(n+k) \in O(n)$, since k is a constant given

Problem with $countsort$ that seems violate Information Theory

It is not a comparison based alg

It solves only a restricted version of sort (we know the range)

Therefore, when solve problems, we should take care of the class of problem and the restrictions.

Adversary arguments

Example a machine will make an integer in $[1, n]$ and let you guess the number is, it will answer too hight or too low for each guess until you make the correct answer. How many guesses will you make to find the correct answer?

Consider a sneaky machine, which picks output depends on your guesses, which answers in a manner that is consistent to the answers before, so that you can't prove it's cheating.

Claim $\forall k \in N$. If the range contains at least 2^k numbers, then you must make $\geq k+1$ guesses.

Proof induction on k

Suppose $k = 0, 2^0 = 1 \Rightarrow$ you must make at least one guess

Suppose $k > 0$, if you answer is $< 2^k$, then the machine will say too low, and removes at most $2^k - 1$ elements, similarly, the machine will remain $2^k + 1 > 2^k$ elements, by induction hypothesis, takes $\geq k+1$ guesses, hence a total of $k+2$ guesses.

Adversary arguments method

$\forall A\ sovles\ P. \forall n\ be\ the\ input\ size$, an adversary can choose an input of size n on which A must take at least $L(n)$ steps.

On each interval node, the adversary will give the edge in a 'sneaky' manner

Example 1 MAX using only $<$

Input: $A[1..n]$

Output $i: A[i]$ is the max

Claim $C(MAX) \geq n-1$

Suppose $\exists T, h(T) \leq n-2, T$ solves MAX

Adversary strategy

Keep track of which elements have lost comparisons and the number c of each elements.

Also, assign distinct values to these elements, initially $c = 0$

Consider a comparison $A[i] < A[j]$
If both have lost, gives the answer according to their given value
If either have lost, answer it's lost again
If none lost, the adversary will say T, assign $A[i] \leftarrow c$, and $c++$, $A[i]$ hence is assigned to be lost

Since $h(T) \leq n-2$, there are at most $n-2$ comparisons when a leaf is reached.
Let i be the return value,
If $A[i]$ has never lost, the adversary will assign it to be $n-1$, and there are $\leq n-1$ elements that have been assigned values.
While there are n elements, hence $\exists j, A[j]$ is not assigned value,
Make $A[j] = n, A[i] < A[j], T$ is incorrect and the adversary is consistent to its previous answers
Contradiction

Example 2 U: searching an unsorted list using ==

Input: $A[1..n], x: key$
Output: $0: x \notin A; i: x == A[i]$

Claim $C(U) \geq n$
Suppose $\exists T, h(T) \leq n-1, T$ solves U
Adversary strategy

Say F all the time

Since $h(T) \leq n-1, \exists j$, $A[j]$ is not compared
If T returns the leaf $== j$, then adversary says $x \notin A$
If T returns 0, then adversary says $x = A[j]$
Contradiction

Reduction

P **reduces** P' if $x \to A \to^{x'} S' \to^{y'} B \to y$
Say A takes $g(n)$ time to map n elements to $f(n)$ elements
B takes $h(n)$ time to map back
$S'WC \in T'(n)$
Hence $C(P') \leq T'(n) \Rightarrow C(P) \leq T'\big(f(n)\big) + g(n) + h(n)$
Contrapositive $\equiv C(P) > T'\big(f(n)\big) + g(n) + h(n) \Rightarrow C(P') > T'(n)$