O(N+M) time complexity

Asked 7 years, 6 months ago Modified 2 years ago Viewed 34k times

25

I'm working through some practice problems where I'm given a target time complexity and space complexity. One of them gives a target time complexity of O(N+M). I'm having some trouble with the intuition of what an O(N+M) algorithm would look like. Does anyone have an example of an algorithm like this or can explain it clearly? Every example I try to think of seems like O(N*M) to me.

algorithm time-complexity

4

Share Improve this question Follow

edited Sep 11, 2014 at 20:27

asked Sep 11, 2014 at 20:18

user137717

1,665 4 20 43

1 Can you post an example of a case with O(N+M)? – Javi Sep 11, 2014 at 20:19

1 Yes, so we can figure out what N and M are in an specific case. All I can think about right now is something like: find the minimum of 2 vector, once of size N and the other one of size M, so that it would be O(M+N) – Javi Sep 11, 2014 at 20:21

1 I'd like to avoid it if I can. The answer below is a good example and I don't want to post the question for fear of someone giving too much of the answer away. Thanks for your help! – user137717 Sep 11, 2014 at 20:31

1 In case you want the formal definition: en.wikipedia.org/wiki/Big_O_notation#Multiple_variables – Tavian Barnes Sep 11, 2014 at 20:50

6 Answers

Active Oldest Votes

32

A simple example of algorithm that is `O(m+n)` :

```
int sum(int[] nArr, int[] mArr) {
    int sum = 0;
    for(int i : nArr) {
        sum += i;
```

```
        }
        for(int i : mArr) {
            sum += i;
        }
        return sum;
    }
```

To compute the sum, you need to go through all elements in `nArr` (size `n`) and all elements in `mArr` (size `m`), so the overall complexity is `O(m+n)`

Share Improve this answer Follow

answered Sep 11, 2014 at 20:24

Jean Logeart
50.3k 11 80 116

1 Complexity-wise, that is impossible to do better :) – Jean Logeart Sep 11, 2014 at 20:27

2 Let's say array nArr with size N is larger. Would it still be correct to say that this algorithm is O(N)? – user137717 Sep 11, 2014 at 20:30

4 No this is not correct, unless `m` is constant or unless `m` depends on `n` such that `m = O(n)` – Jean Logeart Sep 11, 2014 at 20:31

2 Is it correct to think about it as an O(n) algorithm followed by an O(m) algorithm and together they are O(n+m) and abstract this to an arbitrary number of arrays? – user137717 Sep 11, 2014 at 20:41

9 Complexity annotation (such as `O`) describes the way an algorithm's performance (e.g. time or space) is dependent on the size of the input. It is "blind" to a specific input. When you write `O(m+n)` it means that the algorithm will take `O(m)` time when m>n and `O(n)` when m<n. In any way, `O(m+n) = O(max(m,n))`. – Shaked Sep 11, 2014 at 20:49

18

Quick and simple example of an O(n + m) algorithm:

```
for (i = 0; i < n; i++)
{
  // do something but don't loop or invoke recursive functions
  // only constant O(c) complexity is allowed: a simple series of commands
}

for (i = 0; i < m; i++)
{
```

```
  // idem
}
```

Complexity is commutative when added (O(n + m) == O(m + n)) this means you may invert the two `for()` without affecting complexity. Obviously, on an algorithmic level the inverted one *MAY* not be equivalent to the straight one.

As an additional help, here is an example of O(n * m) algorithm:

```
for (i = 0; i < n; i++)
{
  for (j = 0; j < m; j++)
  {
    // do something but don't loop or invoke recursive functions
    // only constant O(c) complexity is allowed: a simple series of commands
  }
}
```

Again, you may invert inside with outside loop without affecting complexity (O(n * m) == O(m * n)). The same obvious considerations apply.

The limitation on what you may put into the `for()` bodies is because the big-o notation constraints the upper bound. If it were a lower bound (little-o notation) you may have put more complex stuff in, but it could never get less than that.

Share Improve this answer Follow

edited Sep 11, 2014 at 21:19

answered Sep 11, 2014 at 21:12

pid
11.2k 5 35 58

2

So, in order to expand the other replies, I will try to add a example of such problems to help you understand:

- Find a min/max in a N sized array, and then look for this value in an M sized array. Since you need to perform first min/max search, you cannot do it at once.

For instance, summing up the elements of 2 vectors can be done in O(M+N), but it can be thought as O(N) (assuming N>M) or O(M) (if M>N).

Share Improve this answer Follow

edited Sep 11, 2014 at 21:01

answered Sep 11, 2014 at 20:56

Javi
3,114 4 25 41

2 Assuming N<M, it is `O(M)`, so `O(M+N)` is simply `max(O(M), O(N))`, which is still linear. – Mephy Sep 11, 2014 at 21:00

1 Well I think that is kind of obvious. However, I have updated the answer. O(M+N) is linear, no matter what the problem is. – Javi Sep 11, 2014 at 21:02

2

All Above Answers illustrate how O(n+m) works but I would like to look at it from a different view by knowing what O(n*m), what is the difference between O(n+m) and O(n*m) the main difference is when multiplying an n number by m number it means that n will happen for m times or will try it for m times, for example, the below code is O(n*m) because n will happen m times for n times

```
for(int i=0; i < n;i++){
  for(int j=0; j < m;j++){
  //some_code
  }
}
```

Share Improve this answer Follow

edited Feb 22, 2020 at 16:15

Mort
1,321 3 16 47

answered Feb 22, 2020 at 15:19

m.mashaly
81 6

1 Doesn't this answer already cover this? – kaya3 Feb 22, 2020 at 15:26

Oops didn't see it – m.mashaly Feb 23, 2020 at 16:56

1

The intuition of this problem is that you have two unique variables `n` and `m`. Now imagine these two unique variables independently increasing, approaching infinity.

If this were an O(n) problem (i.e. BIG-O), the upward boundary of the complexity of this problem would be linear at least. You could say that `O(n) = n^2`. But an O(n) problem would never even get close to that `n^2` limit as `n` (the input) approaches infinite.

Likewise, the behavior for `m` would be the same. `O(m)` can be `m^2`. But it is more accurate to say that `O(m) = m`. The complexities of these two problems are **linear**.

Now, if you just do `O(n+m)`, is that really `n^2`? It shouldn't be. Even if `n=m`, the sum would be `2n` or `2m`. The complexity of this problem is still **linear** because the size of the output is still **proportional** to the inputs `n` and `m`. Therefore, *the most precise answer to this problem* would be `O(n+m) = n+m`.

Share Improve this answer Follow

answered Sep 11, 2014 at 20:48

Caleb Faruki

2,374 3 27 50

1

One instructive example which does something non-trivial is to take two sorted arrays of size M and N and output a new sorted array with all of these elements. This is the basis of merge-sort and will take O(M+N) comparisons.

You can find an example anywhere or do it yourself.

Share Improve this answer Follow

answered Sep 12, 2014 at 0:50

user1952500

6,265 3 22 36