Binary Search Applications

Alex Chen

October 21, 2011

1 What Is a Binary Search?

Problem: Given a list of N unique numbers and Q queries, find out whether each query is among the list of N numbers.
Analysis: The *naive* solution is to use a *linear* search, where, for each query K, we check each of the N numbers to see if K is one of them. This $O(N)$ algorithm will often be too slow. We could use complicated data structures such as a set or a hash table, but a much easier solution is to implement a *binary search*.

A binary search is a *divide-and-conquer* algorithm. At each iteration, we divide the list in half and search from only half of the list for future iterations. How do we accomplish this? First, we must sort the list of N integers. Then, at each step, we check the median of the N integers (the number with the middle index). If this number is greater than the query, then we know to only search in the left half of the list. Else, we search in the right half.

Pseudocode:

```
def binarysearch(list, target):
    sort(list) # if not already sorted
    low = 0, high = len(list) - 1
    while low != high:
        mid = average of low and high
        if list[mid] > target:
            high = mid - 1
        else if list[mid] < target:
            low = mid + 1
        else # we found it!
            return true
    return whether list[low] or list[high] is target
```

The code for a binary search is fairly simple. C++ STL implements it within "algorithm," but for many applications, you will have to implement the binary search yourself.

Complexity: At each iteration, we divide the list in half. We can divide the list in half at most $O(\log N)$ times, so the complexity of a simple binary search is $O(\log N)$.

2 Applications

Binary search techniques are very useful, although not usually obviously, for many problems. These problems must all satisfy one rather simple property.

Find the answer to $P \rightarrow$ Given X, is the answer less than or equal to X?

The above transformation is able to transform many complex problems into binary searches. These problems satisfy two constraints:

1. If X is a possible answer, then one of the following must be true:

- All $y > x$ is a possible answer.
- All $y < x$ is a possible answer.

2. We can check whether x is a possible answer to the problem "efficiently."

If you have a problem that satisfies these two constraints, then chances are you should consider using a binary search to solve the problem.

3 Warm-Up Problems

1. Given an integer N, compute floor($\sqrt{N}$).
2. I am thinking of a positive integer and for each guess, I will tell you whether the number is too high or too low. There is no upper bound on my number. Guess my number in $O(\log N)$ guesses, where N is my number.
3. I am thinking of a positive real number between 1 and N. Guess my number to within one decimal place in $O(\log \log \log N)$ time.

4 The Maximum Average Subsequence Problem

Problem: Given a list of N integers, find the consecutive subsequence of at least length K with the maximum average.

Naive Solution: Check all subsequences of length $\geq K$ within the sequence and find the one with the maximum average. This takes $O(N^3)$, with $O(N^2)$ to check all subsequences and $O(N)$ to average each subsequence.

Quadratic Solution: We can average each subsequence in $O(1)$ time by using prefix sums. We will discuss these in a later lecture, but feel free to look them up or ask. Prefix sums allow us to sum any subsequence in $O(1)$ time, which lets us find the averages. Only $O(N)$ preprocessing is required.

Slightly Faster: Here is a pretty tricky observation. What is the maximum length of the maximum average subsequence that we are looking for. I claim that the solution sequence must be $K \leq L < 2K$. That is, the upper bound on the length of the maximum average sequence is $2K$. **Why?** Now, we can just check all sequences of lengths between K and $2K$, for a complexity of $O(KN)$. Unfortunately, K can be up to N so this is still quadratic.

4.1 Using a Binary Search

Let's check our two constraints to see if we can use a binary search for this problem. We are looking for the maximum average. This is the answer X that we want. We can transform this problem:

1. If there is a subsequence of at least average X, then there must also be subsequences with at least average Y for $Y < X$.
2. We can check whether there exists a subsequence of at least average X in $O(N)$ (this is not immediately obvious).

Point 1 is fairly straightforward, because the same subsequence that has average X must also satisfy all $Y < X$. Now we have the following pseudocode for this problem:

```
def solve(list):
    low = 0, high = maximum possible average
    while low < high:
        mid = average of low and high
        if there exists a subsequence with average >= mid:
```

```
            low = mid
        else:
            high = mid
    return average of low and high
```

The main tricky part is checking whether there exists a subsequence with average at least mid. Recall that this subsequence must have length at least K.

4.2 The Checking Algorithm

For binary search problems, the binary search is not the hard part. Often, it is the checking part. We can use some mathematical manipulations to solve this. Let A be the average and C_i be the i^{th} element of some subsequence of length n. **We want to answer the question: does there exist a subsequence with average that is *at least* A?**

$$\begin{aligned} A &\leq \frac{1}{n}\sum_{i=1}^{n} C_i \\ nA &\leq \sum_{i=1}^{n} C_i \\ 0 &\leq \left(\sum_{i=1}^{n} C_i\right) - nA \\ 0 &\leq \sum_{i=1}^{n} (C_i - A) \end{aligned}$$

This is now the maximum subsequence sum problem for a different sequence. We want to find whether there exists a consecutive subsequence, where each element is $C_i - A$, with a nonnegative sum. This is a fairly tricky problem in and of itself, and we can solve this using prefix sums as well. Something that helps is that we can simply find the maximum subsequence sum and check if it is nonnegative. Complexity: $O(N \log N)$.

5 More Problems

For each of these problems, try and determine the "inner problem" involved (the problem to check) and the target of the binary search (what variable are you binary searching on?).

1. Find the largest palindromic substring of a string in $O(N \log N)$ (this can actually be done in linear time!).
2. (Jan Kuipers 2004) Farmer John has built a long barn, with N ($2 \leq N \leq 100,000$) stalls. The stalls are located along a straight line at positions $x_1, \ldots, x_N$ ($0 \leq x_i \leq 1,000,000,000$). His C ($2 \leq C \leq N$) cows want to be as far away from each other as possible. Farmer John wants to assign all C cows to one stall each such that the minimum distance between any pair of cows is maximized. Find this minimum distance.
3. (Mihai Strohe, 2005) Farmer John has a farm that is a grid of $N \times N$ cells, each of which has an altitude ($0 \leq A_{i,j} \leq 2,000,000,000$). Farmer John starts at $(1, 1)$ and wants to reach (N, N). He may only travel up, down, left, and right. Due to altitude sickness, Farmer John wants to minimize the value of the maximum height he encounters along his path. Find this minimum height.
4. (Brian Dean & Andre Kessler, 2009) Farmer John's cows are going river rafting. The riverbanks are two parallel lines. The river contains many rocks, each of which is a circle. No two rocks overlap. Find the radius of the largest raft (also a circle) that the cows can ride down the river without going through any rocks.