Greedy Algorithms

Context In general, when you try to solve a problem, you're trying to find or build a solution among a large space of possibilities. You usually do this by making a series of decisions, what move to make at each step (for example: send 2 cannibals across first, or 1 cannibal and 1 missionary, etc.).

If you have no information or no way to tell which choice is best, then you may have to do an exhaustive search over all the possibilities using backtracking. Basically this means you pick some choice and go down that path until you either reach a solution or hit a dead end, in which case you go back (undo) and try something else. While this will work in principle, in practice it's often unusable for realistic problems because there are just too many possibilities to explore. Even relatively simple-seeming problems may have billions upon billions of possible solutions, and it would take infeasibly long to check them all.

Greedy Algorithm In some cases, the problem has some underlying structure that lets you be more intelligent, and you can figure out the right choice at each step, without ever needing to undo or backtrack a decision. In the happiest cases, when you're especially lucky, there is sufficient structure that you can quickly reach a solution by just picking the straightforward "best" choice at each step. This is called the Greedy Algorithm.

The bad news is that greedy doesn't always work! When it does work it's great, but for many problems, when you pick what looks like the best choice for part of the solution, you can later find that it actually leads you down the wrong path and forces bad choices for other parts of the solution. So before you apply Greedy, it's important to prove that it actually will find the best solution for the problem you're trying to solve.

Another point to mention here is that even for a single problem, there may be more than one potential greedy strategy, more than one way to determine what looks like the "best" choice at each step. And it could be that Greedy works using one strategy but not using another strategy. So you need to prove that it works for the particular strategy you're using.

Other Approaches In cases where Greedy doesn't work, you still may be able to use other approaches that are much better than doing an exhaustive search, as long as there is some structure to the problem. One such approach is Dynamic Programming, which will be covered next week. Incidentally, in a way you can think of Greedy as a simple special case of Dynamic Programming.

1 Problems

Problem 1: Lemonade Stand Placement

Five friends – Alex, Brenda, Cathy, Dan, and Earl – want to set up a lemonade stand. They live at the locations denoted by the circles on the map below. At which street intersection should they place their stand to minimize the total distance to their homes? Assume that they measure the distance by the total number of blocks – horizontal and vertical – from their homes to the stand.

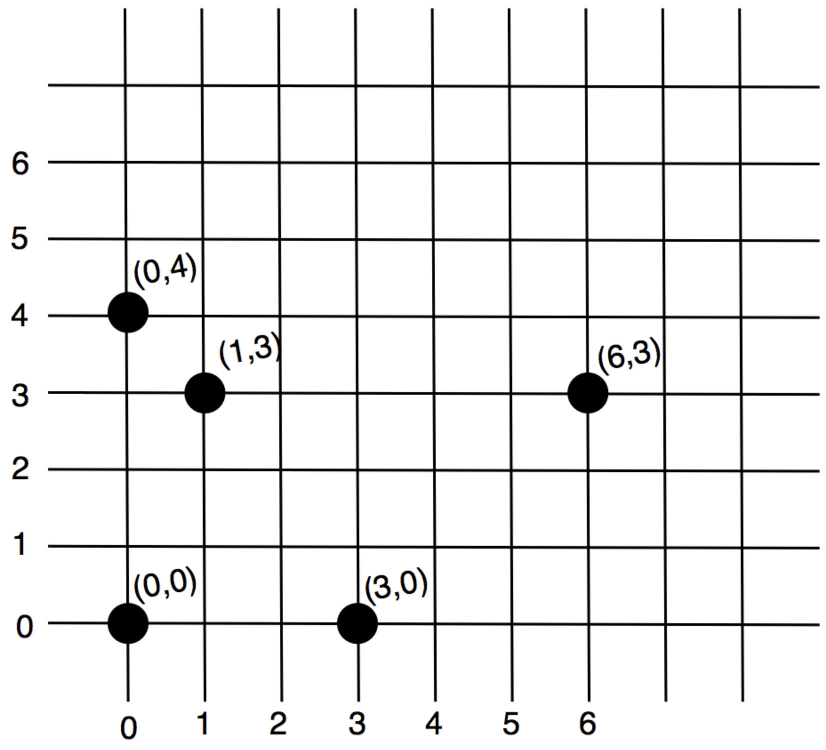

Figure 1: Lemonade Stand: locations of houses.

Solution: Lemonade Stand Placement

First, note that you *could* do an exhaustive search, trying out all possibilities. But there are a lot of possibilities. For this problem it's small enough that we could do it, but I could give a much larger example that's more spread out where there would be too many.

But we're lucky, in that this problem has a nice structure that allows us to essentially jump straight to the solution by doing a greedy approach:

The first thing is to decide how we will make a choice at each step. What we'll do is first choose the East-West position, then choose the North-South position. (We could just as well do North-South first, it doesn't matter). It may not be obvious at this point that solving East-West and North-South independently is valid, but we'll do this for now, and then at the end prove that it leads to the right solution.

Brainstorming

So, how do we make the best choice for the East-West position?

- average the positions of the houses? The average of the houses' East-West positions is $\frac{0+0+1+3+6}{5} = 2$. The total (East-West) distance from this to the houses is then $2+2+1+1+4 = 10$. Is this the best we can do? Let's do some sanity check to see if we could improve upon this. What if we move the lemonade stand one spot to the West, to position 1? Then the total distance is $1 + 1 + 0 + 2 + 5 = 9$. That's better than what the average position gave us, so average is not a good strategy here — it can lead to the wrong answer.

- take the center of the range? The positions range from 0 to 6, so the center of this is position 3. The total (East-West) distance from this to the houses is $3 + 3 + 2 + 0 + 3 = 11$. This is worse than the average method, so this can't be right.

- the median, or middle house? The median house is the one "in the middle", if you line them up in order. It has the same number of houses on either side of it. In this case, the median house is the one at position 1. The total distance to this house is $1 + 1 + 0 + 2 + 5 = 9$. This is better than the above strategies. (In fact, this was the example position we used to show that average is not valid.) Is this one guaranteed to be the best? Let's go with it for now.

Then do the same for North-South, taking the median of the houses' North-South positions. For our map, the North-South positions are 0, 0, 3, 3, and 4. The median of this is 3, so that will be the North-South position of the lemonade stand.

Putting these together, the lemonade stand will be at 1 East and 3 North. (This happens to be where one of the houses is, but that won't necessarily always be the case if the houses are set up in different places.)

Validate the Solution

Now, we do have to prove that this is the best solution. Remember that average and middle of range don't work. There's no reason to assume median works — it just happens to be the best strategy we looked at. But in fact we can prove that it will always find the best location.

Here's how we'll do it: say we take our median location above. Can we do better? Let's say we can, and let's see what happens when we move from our location to the better location. For each step we take East (or West, North, or South), we move one step towards all the houses on the East side, and move one step away from all the houses on the West side. Since there are the same number of houses on each side (since we started at the median), these changes balance out, and the total distance doesn't change[1]. But, we also move away from the median house, which was at the same location as the lemonade stand, so the distance to that house increases by 1, and so the total distance increases by 1. Furthermore, we now have more houses to the West than to the East, so each additional step East moves away from more houses than it moves toward, increasing the total distance even more. In sum, if we move East from the median location it will always increase the total distance. And of course the same argument applies if we instead move West, North, or South from the median location. Therefore the median location is indeed the best, and this is a valid Greedy strategy.

Variant

This problem takes place in 2 dimensions: East-West and North-South. What if we lived in a city of the future where all the buildings are skyscrapers and there are "up-down blocks" as well,

[1]One technical detail: some of the other houses may have the same East-West position as the lemonade stand itself. In that case the changes in distance may not balance out. But even in this case it's guaranteed that the total distance will stay the same or increase.

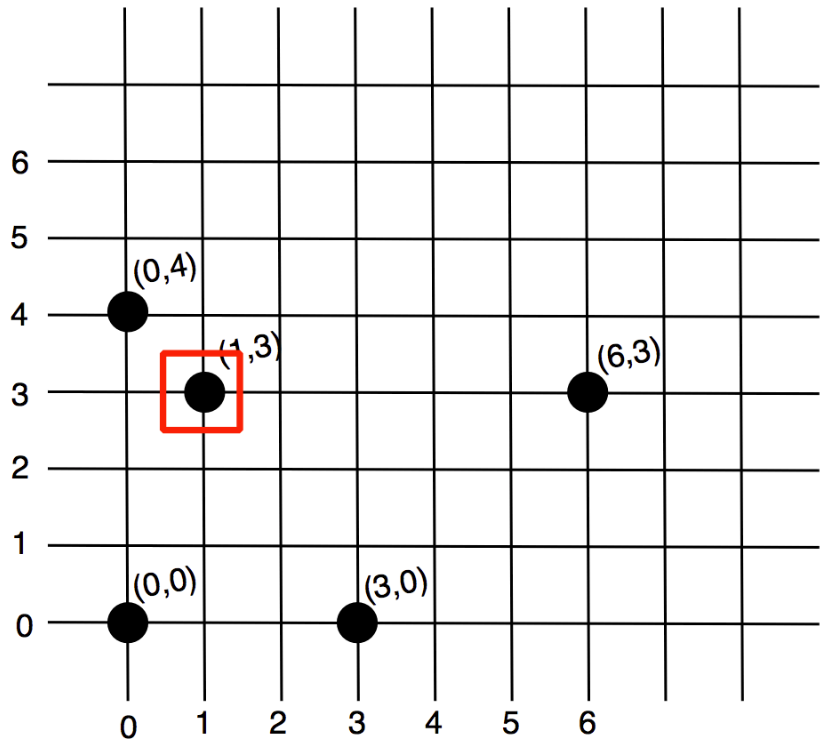

Figure 2: Lemonade Stand Solution: The best location for the lemonade stand is indicated with a red square.

and all buildings have sky bridges at regular intervals so you can switch buildings at whatever height you're at, without needing to go down to the ground level. Now the lemonade stand and each person's apartment has a height in addition to its East-West and North-South coordinates. Can you use the same algorithm? How would you adapt it? How many steps would it take?

Reflections

This Greedy Algorithm takes 2 steps, one to determine the East-West position, and one to determine the North-South position. No matter how many houses there are, or how big the city is, it takes two steps. This is in sharp contrast to exhaustive search, which has to examine more and more possibilities the larger the city is. This is typical of greedy approaches; the number of steps is related to how "big" the solution is (in this case, 2 dimensions) rather than how big the problem is.

Problem 2: Heads Up

There are n coins in a line, heads and tails in random order. On each move, one can turn over any number of coins laying in succession. Design an algorithm to turn all the coins heads up in the minimum number of moves. How many moves are required in the worst case?

For a concrete example, consider the case where there are 7 coins, initially in the configuration THHTTHT.

Solution: Heads Up

Brainstorming

How can we approach this? What are some ideas:

Most obvious thing is to just take each separate group of Ts and flip them, one group at a time. Then the number of flips required is the number of groups of Ts. Is this the best we can do? How many moves would be required in the worst case? What is the worst case? Consider this a baseline as we think of other approaches. THHTTHT $\rightarrow$ HHHTTHT $\rightarrow$ HHHHHHT $\rightarrow$ HHHHHHH. 3 steps.

What other ideas are there? Could it ever help to flip an H to a T? It seems like that would be going backwards.

What if we flip as many Ts as we possibly can, even if it means turning over some Hs? Then: THHTTHT $\rightarrow$ HTTHHTH $\rightarrow$ HHHTTHH $\rightarrow$ HHHHHHH. Still 3 steps. But maybe for some problems this would be a better approach?

Or, maybe we should be wary of turning over too many Hs. Maybe we should make the move that leads to the most total number of Hs after each step: THHTTHT $\rightarrow$ THHHHTH $\rightarrow$ HHHHHTH $\rightarrow$ HHHHHHH. Still 3 steps for this, but looks like it made more progress after 1 step. Maybe this is better for large complicated examples?

How would this do in the worst case? What is the worst case? Well, a bad situation is if you never have 2 Ts in a row, so you can't make a move that gains a lot of Hs without losing almost as many. Alternating THTHTHT seems like the worst for this. It turns out that's also the worst for the other approaches above.

Following this line of thought, maybe we should try, on each step, to reduce the numbers of "transitions" between H and T (or vice versa). We want to get it more "smooth" and less alternating. With that in mind, what's another way to think about what we're doing with a single move? Ignore how many Hs and Ts for now, just think about the transitions. At either end of the group we're flipping, we are either creating a transition or removing one. Our goal is to have all H, which has 0 transitions, so maybe we should just focus on reducing transitions on each step.

Now this is something we can analyze. We could count how many transitions there are at the beginning. For example in HTTHH there are 2 transitions. At each step, we can reduce it by 2, increase it by 2, or leave it the same. Of course we want to reduce it by 2. When there are 0 transitions left we're done. So the number of steps needed is half of the initial number of transitions.

One problem with the above analysis is that TTTTTTT also has 0 transitions. How do we avoid ending up with all Ts? Just define a T on either end of the line to count as one transition. (You can think of it as if the whole string is bookended by Hs on either end that you're not allowed to touch, but still count for transistions, so TTTTT is like H:TTTTT:H).

Then the only way to get to 0 transitions is to have all Hs. In our original example, THHTTHT, there are 6 transitions at the beginning, and so it must take at least 3 steps to get to 0 transitions.

Now looking back at the approaches we brainstormed, it turns out that all of them reduce the number of transitions by 2 on each step, so all them are optimal and take the same number of steps, even though they all do it in different ways. Any approach that chooses sequences on transition boundaries will achieve the smallest number of moves.

One final detail we need to show to complete the proof. We need to show that no matter what configuration the coins are in, we can always find a move that reduces the number of transitions by 2. For example, could there be some weird configuration where there is only 1 transition? The answer is no: no matter what, we can go along until we find the first T. That must come right after a transition, since it either follows an H or is the first coin in the line. Then we can choose the last T (even if it's the same one) for the other end of our sequence. And likewise, this must have a transition right after it. So we can always find a move the reduces the transitions by 2.

Just as an aside: from the above we can conclude that the starting position (or any position) must have an even number of transitions, since we know we can keep reducing by 2 to get down to 0. This isn't part of the proof that the algorithm works, just an interesting note.

Variant

What if coins are in a loop instead of a line?

Reflections

Brief summary of approach: reduce transitions by 2 each time. Can ignore how many Hs and Ts at any time. Guaranteed to get down to 0, and can't do it any faster. As with many greedy approaches, the easy part is finding a solution, but the key is to prove that it works.

All of the approaches we brainstormed take exactly the same number of steps, even though they do different things and seem to have different levels of sophistication. So which approach should you use? One consideration is how hard it is to figure out what flip to do in each step. The baseline approach can just go along and pick the first T and then the next H. That's very simple to calculate. The most "sophisticated" approach looks at possible flips and picks the one that leads to the most Hs. In practice that will take much longer to run, and be harder to write, even though they make the same number of flips in the end. So in this case the simplest approach is also the fastest in practice, and that's what we'd choose.

2 Homework

Problem 3: Averaging Down

There are 10 identical vessels, one of them with 100 pints of water and the others empty. You are allowed to perform the following operation: take two of the vessels and split the total amount of water in them equally between them. The object is to achieve a minimum amount of water in the original vessel (the one containing all the water in the initial setup) by a sequence of such operations. What is the best way to do this? You can do as many operations as you want — the goal is just to minimize the amount of water in the original vessel.

3 Advanced Problems

Problem 4: Rumor Spreading

There are n people, each in possession of a different rumor. They want to share the news with each other by sending electronic messages. What is the minimum number of messages they need to send to guarantee that every one of them gets all the rumors? Assume that a sender includes all the rumors he or she knows at the time the message is sent and that a message may only have one addressee.

Problem 5: Bachet's Weights

Find an optimal set of n weights w_1, w_2,... , w_n so that it would be possible to weigh on a two-pan balance scale any integral load in the largest possible range from 1 to W, assuming the following:

(a) Weights can be put only on the free pan of the scale.

(b) Weights can be put on both pans of the scale.