Algorithms and competitions

3 Strategies: Decrease and Conquer, Divide and Conquer, Transform and Conquer

Victor Liew Jia Hao

Decrease and Conquer strategy is used if we could reduce a problem into its smaller state by a factor and solving for it. The solution to this smaller instance, would form the basis to larger instances of the same problem.

Problem of size n

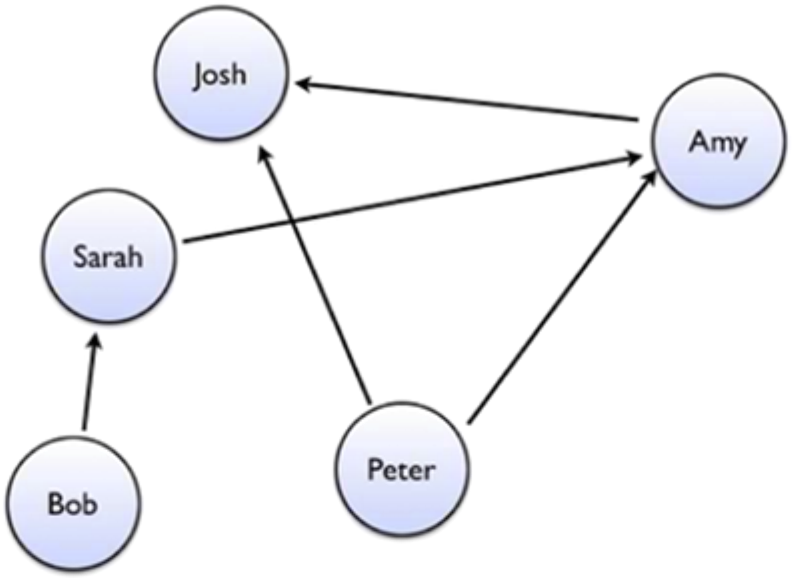
Solve for n - 1 (or n - constant factor)

Repeat the above and eventually the problem is reduce to something trivial that we can solve immediately. ie: n = 1 or n = 0

Celebrity problem: In a party of N people, only one person is known to everyone. Such a person **may or may not be present** in the party. The celebrity doesn't know anybody.The task is to find the celebrity, but we can only ask questions like "**does A know B?"**

```
#!/usr/bin/perl

#suppose we have 10 people in this room
my @party = ('ppl',
'ppl','ppl','ppl','ppl','ppl','celeb','ppl','ppl','ppl','ppl');

#repeat till there's only 1 person left, which by defn, makes this person
a celebrity
while($party > 1) {
      #Pick 2 person, ask A if he knows B. If No, B isn't celebrity. If
Yes, A isn't celebrity. (hence decrease by 1 possibility)
      my $A = shift @party;
      my $B = shift @party;
      my $AknowsB = isKnow($A,$B); #assume isKnow has been defined

      if($AknowsB) {
            push $B @party; #Add B back into room, while discarding A as
a possibility
      } else {
            push $A @party; #Add A back into room, while discarding B as
a possibility
      }
}

#answer is found.
my $solution = shift @party;
```

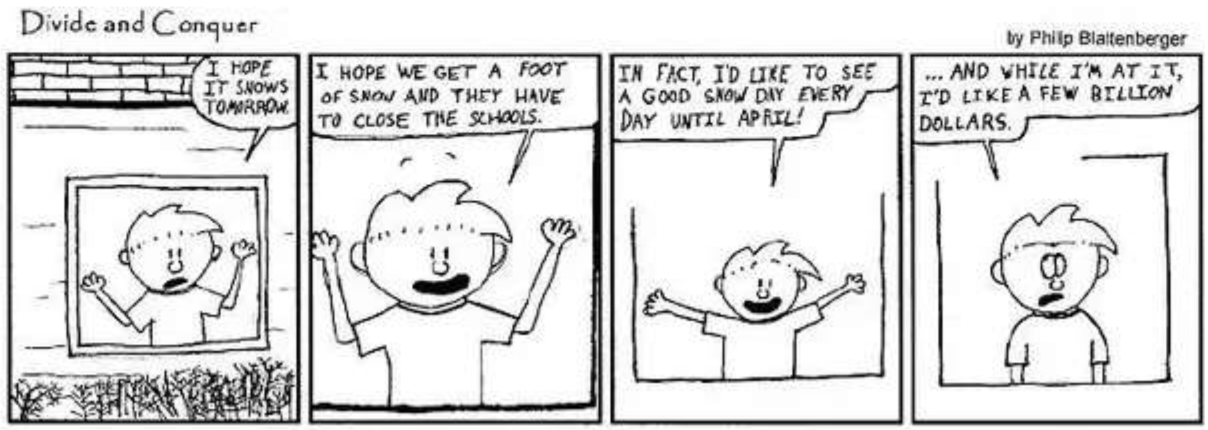
The above solution might look trivial. But suppose we do not know about



the Decrease and Conquer strategy, and choose to use **Graphs as our algorithm** instead, which proves to be inefficient:

We can model the solution using graphs. Initialize indegree and outdegree of every vertex as 0. If A knows B, draw a directed edge from A to B, increase indegree of B and outdegree of A by 1. Construct all possible edges of the graph for every possible pair [i, j]. We have NC2 pairs. If celebrity is present in the party, we will have one sink node in the graph with outdegree of zero, and indegree of N-1. We can find the sink node in (N) time, but the overall complexity is O(N^2) as we need to construct the graph first.

Source: The Celebrity Problem - GeeksforGeeks


Divide and Conquer
by Philp Blaltenberger
I HOPE IT SNOWS TOMORROW.
I HOPE WE GET A FOOT OF SNOW AND THEY HAVE TO CLOSE THE SCHOOLS.
IN FACT, I'D LIKE TO SEE A GOOD SNOW DAY EVERY DAY UNTIL APRIL!
... AND WHILE I'M AT IT, I'D LIKE A FEW BILLION DOLLARS.

Divide and Conquer strategy is used if we could break a larger problem into several smaller subproblems. These sub problems could then be combined if necessary to obtain the solution to the larger problem.

Divide and conquer VS Decrease and conquer:

- Decrease and Conquer involves reducing the problem into 1 sub problem, while divide and conquer reduces the problem into several subproblems.
- ie: Divide and Conquer would yield something along the lines of T(n/x), where x is amount of subproblems. Whereas Decrease and Conquer would yield something like T(n - c) where c is a constant factor.

Transform and conquer involves changing the problem into something that is much easier to solve, or to an instance in which you know the solution to.

- In the celebrity problem, if you are a graph expert, you would have interpreted the problem as a graph problem, and model your solution base on vertices and edges.
- You could also model the problem using arrays, stacks etc. Some representations are inherently easier to digest. (Knowing the right data structure to use is hence crucial)
- You could also simplify a problem first. In the case of a search problem, we could make our lives easier by sorting the array

  A very good introduction to this approach could be found here: Page on Uctleg

Personally, I felt that the transform and conquer approach is one that could be eventually grasped by anyone through loads of practice