Constructive Algorithms

Alex Li

March 23, 2019

1 Introduction

There are 3 main types of algorithms. In competitions, we most often seen algorithms that try to find THE correct answer. However, sometimes this is too hard, and we resort to algorithms about estimating the best answer. Occasionally, however, there are many equally good answers, and we only need to find one. Maybe even, there is no incorrect answer, and it's really about the fact that you tried... ok, no, programming is not so nice. Anyways, sometimes you just need to construct a right answer, and if this is the case, your goal should be to construct the most simple right answer as possible.

2 Problem 1: Nice Table

http://codeforces.com/problemset/problem/1098/B
The first thing to ask in this problem, and indeed with most constructive algorithms, is: what can I make that satisfies the constraint? So we draw a few grids out and try to develop some ideas, for now, not even worrying about optimizing anything. (draw 5 by 5 grid) Notice any patterns? (Oh, we alternate between pairs of 2 letters in each row/col) (And in fact, every valid grid can be generated in this way) Ok, we have really limited the search space. Now what? Since there are only 6 ways to choose a pair of letters, we can fix that. We can also fix the direction. Now, for each row or column, we can just choose the best of the 2 different ways to orient it. (This is not a true constructive algorithm I think because there is only 1 right answer quite often, but it seems like it is at first)

3 Problem 2: Balanced Neighbors

https://atcoder.jp/contests/agc032/tasks/agc032_b There is... just no clear way to start. Well, I guess we can make some examples. (Show solution for n=3, ask for n=4 (try 5?)). Since it's so hard to make solutions, we may be forced to find a pattern in our n=4 solution. Notice that we divide the vertexes into 2 groups - (2,3) and (1,4), forming a complete bipartite graph. Because 2+3=1+4, this

works. Get excited, then realize that there is no way this will work for 5 vertices. But, maybe we can separate into 3 equal groups: $5 = 2 + 3 = 1 + 4$. So, now we just need to divide our graph into groups with equal weight... this is a hard problem, though):. Solution: always do groups that are pairs.