Algo HW1

Kevin Klaben (kek228)

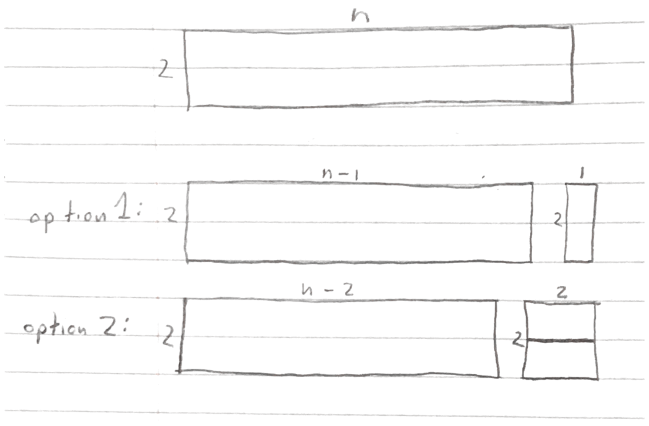
January 29, 2019

(1) For each positive integer n, let tn denote the number of distinct ways to cover a rectangular 2n grid with non-overlapping dominoes. What is the value of tn? Prove the correctness of your answer using mathematical induction.

Solution

Claim:
$t_n = t_{n-1} + t_{n-2}$

Proof:
This proof will be done by strong induction on n. Let P(n)="$t_n = t_{n-1} + t_{n-2}$".
Base Case: The values of t_1 and t_2 must be calculated as t_0 does not exist. This is done simply by inspection and it can be seen that $t_1 = 1$ and $t_2 = 2$. Thus, the first value for which the claim can be made is t_3. We want to show that P(3) as our base case thus that $t_3 = t_1 + t_2$. As stated above $t_1 = 1$ and $t_2 = 2$ and by inspection there are 3 ways to arrange dominoes to fill a 2x3 rectangle thus as 3=2+1 the claim holds in the base case.
Inductive Step: We want to show that P(n) holds given P(n-1)...P(1) thus that $t_n = t_{n-1} + t_{n-2}$. We have a 2xn rectangle to be filled and this is equivalent to a 2x(n-1) rectangle with any additional domino in the upright position added on the end shown labelled option 1 in the image below. The only other possibility that this expansion over by one makes possible is that the 2xn rectangle is equivalent to a 2x(n-2) rectangle with two stacked horizontal dominoes occupying an attached 2x2 rectangle shown labelled option 2 in the image below.



We know that there are t_{n-1} distinct arrangements of dominoes for a 2x(n-1) rectangle and we also know that there are t_{n-2} distinct domino arrangements for a 2x(n-2) rectangle. Together these two cases represent all possible distinct domino arrangements for a 2xn rectangle and thus put together they total t_n thus it holds that $t_n = t_{n-1} + t_{n-2}$.

(2a) An explicit input on which the algorithm outputs the wrong answer is as follows:

Employer preferences:	Applicant preferences:
$y_1 : x_1 > x_2 > x_3$	$x_1 : y_2 > y_3 > y_1$
$y_2 : x_3 > x_2 > x_1$	$x_2 : y_1 > y_3 > y_2$
$y_3 : x_1 > x_3 > x_2$	$x_3 : y_1 > y_3 > y_2$

When the algorithm is run starting with y_1 with the exclusion of y_3 and x_3 we first have the pairing of (y_1, x_1). Next, (y_2, x_2) becomes a pairing according to their preferences. Now according to the algorithm the pairing of (y_3, x_3) is introduced and in this case the answer no is produced as (y_3, x_1) forms an unstable pair as they each prefer each other to their current matchings. However, with these preferences a stable perfect matching does in fact exist as the set [(y_1, x_2), (y_2, x_1), (y_3, x_3)] is a stable perfect matching as it has no unstable pairs.

(2b)

To modify the G-S Algorithm for this solution we can simply create a Forbidden set to distinguish which pairs are forbidden to be made. The forbidden set will include all pairings of applicant x_n with every employer above y_n in the preferences of x_n. Similarly, all pairings of employer y_n with every applicant above x_n in the preferences of y_n. In addition, for each applicant above x_n in the preferences of y_n, the employers in their preferences below y_n and that applicant will be added to the forbidden list. The same technique is used to create forbidden pairings for those other employers in the preferences of x_n below y_n. After this, the G-S algorithm will be run with an altered condition, the condition will be run while there is an employer y who is free and has not proposed to each applicant x where the pairing of the (y,x) is not on the forbidden list. If at the end there is an employer left unmatched then the algorithm outputs no meaning that there does not exist a perfect stable matching for the given preferences which includes the pairing (y_n, x_n). If at the end every employer has been paired then this is a stable perfect matching including the pair (y_n, x_n). Formally the algorithm is as follows:

Pre-Processing Step:

1.)Examine the preference list of x_n and add the pairing of (x_n,y) to the Forbidden List in decreasing preference until y_n (the desired pairing is reached) and then stop.

2.) For each y that has no been added to the forbidden list as a pairing with x_n, on their preference lists for each applicant x below x_n the pairing (x,y) should be added to the Forbidden List as well.

3.) Similarly, Examine the preference list of x_n and add the pairing of (x_n,y) to the Forbidden List in decreasing preference until y_n (the desired pairing is reached) and then stop.

4.) For each x that has no been added to the forbidden list as a pairing with y_n, on their preference lists for each employer y below y_n the pairing (x,y) should be added to the Forbidden List as well. [H]

```
Modified G-S Algorithm:
Begin with all employers(y's) and all applicants(x's) free
While there is a free employer y who hasn't proposed to each applicant x
  for which the pairing is not on the Forbidden List
  Choose this employer y
  Find the most preferred applicant x for which y has not already proposed
  to and (x,y) is not on the Forbidden List
    If x is free, (x,y) become a pair
    Else x is already paired to another employer y'
      If x prefers y to y' then
        x switches and pairs with y making (x,y)
        y' becomes free
      Else x prefers y' to y then
        y stays free
      EndIf
    EndIf
  EndWhile
  If there exists an employer that is free then "No" is returned
  Else "Yes" is returned
  EndIf
```

Proof of Correctness:
Let M be the matching generated by the algorithm
Observation 1: Given the algorithm outputs "Yes" then M is a perfect stable matching.
Part 1: The output is yes, then M is a perfect matching.
Proof: By definition the algorithm only outputs "Yes" if every employer has been paired. Each employer can only be paired to one applicant and there are equal number of applicants as there are employers. Thus, as all employers are paired it follows that all applicants are also paired and each can only be paired once, thus M is a perfect matching.
Part 2: The output is yes, then M is a stable matching
Proof: Suppose for the sake of contradiction that there exists and unstable pair (x,y). Then we know that x prefers y to its current partner y' and that y prefers x to its current partner x' and that (x,y) is not currently a pairing in M. Then we know that y must have either proposed to x as it is currently paired with x' or (x,y) must have been an element of the forbidden list. In the case that y proposed to x is would have had to have been displaced by some y" which was higher ranked by x, thus x: y"¿y. However, as x is currently paired with y', x must have preferred y' to y", thus x: y'¿y". This a a contradiction as this means that x: y'¿y and out assumption that (x,y) was an unstable pair means that x: y¿y'. The case that (x,y) is a forbidden pair is in fact impossible. In the case that the unstable pair was put n the forbidden list by pre-processing

step 2 or 4 the pair cannot possible be unstable. As was stated both x and y were previously paired. The forbidden pairs generated by these pre-processing steps are the least desirable matches for those specific x and y and thus they cannot possibly be preferred to their current pairs. Similarly, if the forbidden pair was put on the list by steps 1 or 3 these cannot possibly be unstable as all less x and y which were more desired by y_n and x_n respectively, had all of their less desirable pairings forbidden in steps 2 and 4. As these matches cannot be preferred by both x and y in the pairing and unstable pairing cannot possibly be a forbidden pair. For this reason clearly M contains no unstable pairs.
Observation 2: The output is ”No”, then the matching M is not a perfect stable match. Proof: By definition in the algorithm, ”No” is only output if there is an employer who is left unmatched at the end. This means that by definition the matching M is not perfect and in turn it cannot be a perfect stable matching.

Run time Analysis Pre-Processing Step:
In steps 1 and 3 each preferences could possibly be accessed once in a list containing n elements thus the maximum number of preferences accesses made is n. In steps 2 and 4 which happen for each preference accessed in steps 1 and 3, at a maximum the entire preferences list must be accessed in succession thus there would be n accesses. With the actual additions of pairing to the Forbidden list taking constant time the overall run time for the pre-processing step would be O(n^2)
Run time Analysis Processing:
As at a minimum there may be no forbidden pairs, the algorithm could at a maximum have the same run time as the G-S algorithm. As in the G-S algorithm there are n employers who may potentially have to make a total of n proposals each thus the overall run time is O(n^2).
Run time Analysis Post-Processing Step:
This step is simply checking to see if each employer has been paired, this would at the maximum involved iterating over each of the n employers and in constant time checking if they are paired or not. Thus, the run time for this step is O(n).
Overall Run-Time is O(n^2)+O(n^2)+O(n)=O(n^2)