

1 Reverse Preference Stable Matching

A witch bewitches all the jobs just before a run of the propose-and-reject algorithm, so they propose in reverse order of their preference lists (equivalent to the jobs giving reversed lists as inputs to the algorithm). The spell wears off after the algorithm is run and a matching is obtained. **Only the jobs reverse their preference lists, and after the spell wears off, the jobs remember their original preferences.**

- (a) Find the output matchings for the following two instances.

(i)	j.	pref. ($\succ$)		c.	pref. ($\succ$)	
	1	A	B	A	2	1
	2	A	B	B	1	2

(ii)	j.	pref. ($\succ$)		c.	pref. ($\succ$)	
	1	A	B	A	1	2
	2	B	A	B	2	1

- (b) Prove that for any output of the bewitched propose-and-reject algorithm, the matching is either unstable or all the candidates got their first choice job.

Solution:

- (a) (i) $\{1B, 2A\}$
(ii) $\{1B, 2A\}$

- (b) We want to show that if not all candidates got their first choice job, then the matching is unstable.

Suppose there is some candidate c that is paired with job j but prefers some other job j^* . Let c^* be the candidate paired with j^* . Note the matching is the output of the propose-and-reject algorithm on the instance with reversed job preferences. Thus the matching is stable with respect to the instance with reversed job preferences, so j^* prefers c^* to c . Then in the original preferences, j^* prefers c to c^* , so (j^*, c) is a rogue couple (with respect to the original stable matching instance). Thus the matching is unstable.

2 Error Correction

Suppose we want to send a message of n packets (viewed as an integer modulo some large prime q) across an unreliable link. We anticipate that up to 7 packets may be corrupted, so we send a codeword consisting of $n + m$ points $P(1), \dots, P(n + m)$ on some $\text{GF}(q)$ polynomial $P(x)$ of degree $n - 1$. Answer the following questions about this scenario.

- (a) What value should we choose for m ? No justification required.
- (b) Suppose that in reality 10 packets are corrupted, and we receive the values $r_1, r_2, \dots, r_{n+m}$, where m is as in part (a). It is still possible to recover the polynomial P if we are told the locations of some of the errors. What is the minimum number of error locations we need to know to achieve this? No justification required.

- (c) Let ℓ be the minimum number of error locations you found in part (b), and suppose we are told that those error locations are at positions $1, 2, \dots, \ell$. Briefly explain how you would recover the original message given this information. Be sure to specify the degrees of the polynomials $E(x)$ and $Q(x)$ and the points on which they should be evaluated. Your answer should be in terms of the variables m and ℓ defined in parts (a) and (b), respectively. You may assume familiarity with the Berlekamp–Welch algorithm, but you should indicate how to apply it in this setting.

Solution:

- (a) $m = 14$. As shown in class, if k packets are corrupted we need to send $2k$ extra packets.
- (b) 6. Suppose we are told the locations of ℓ errors. Ignoring those locations, we are left with $n + 14 - \ell$ packets, $10 - \ell$ of which are corrupted. By the same principle as in part (a), we need to ensure that the number of excess packets is $14 - \ell \geq 2(10 - \ell)$, which we can solve to get $\ell \geq 6$. As a check, note that $n + 14 - \ell = n + 8$ and $10 - \ell = 4$.
- (c) As explained in part (b), we ignore the first ℓ packets (which we know are corrupted), leaving us with $n + m - \ell$ packets, $10 - \ell$ of which are corrupted. We run the Berlekamp–Welch algorithm on the remaining packets to recover the original polynomial $P(x)$ as follows. Let $E(x) := (x - e_1) \cdots (x - e_{10-\ell})$ be the error polynomial, where $e_1, \dots, e_{10-\ell}$ are the locations of the remaining, unknown errors; note that $E(x)$ has degree $10 - \ell = 4$ and leading coefficient 1. We now evaluate the (unknown) polynomial $Q(x) := P(x)E(x)$ at the $n + m - \ell = n + 8$ points $x_i = \ell + 1, \ell + 2, \dots, n + m$ using the equations $Q(x_i) = r_i E(x_i)$. This gives us $n + m - \ell$ equations in $n + m - \ell$ unknowns (the coefficients of $P(x)$ and $E(x)$), which we solve as usual to obtain $P(x)$ and $E(x)$. Now that we have $P(x)$, we can recover the original message (including the ℓ corrupted packets whose locations we were given).

Note: Instead of ignoring the first ℓ packets, it is also valid to retain them and to modify the error polynomial $E(x)$ by including the additional factors $(x - 1) \cdots (x - \ell)$ corresponding to the known error locations. The algorithm then works as before, except that the degrees of $E(x)$ and $Q(x)$ both increase by ℓ , and we need to use all $n + m$ points $x_i = 1, \dots, n + m$. This solution is less efficient than the one above, but equally valid.

3 Anna's Blackboard

Anna writes down an arbitrary positive integer n on a blackboard. She then repeatedly performs the following operation, where k is the number currently on the board:

- if k is a multiple of 3 then replace k by $k/3$; else replace k by $k + 1$

Prove that, for any starting integer n , Anna will always eventually write the number 1 on the board.

Solution: Let $P(n)$ denote the statement that, when Anna starts with n on the board, she eventually writes 1 on the board. Our goal is to prove $(\forall n \geq 1)P(n)$. We proceed by strong induction on n .

Base Case: $n = 1$. This is trivial since 1 is already on the board.

Inductive Step: Fix any $n \geq 2$, and assume $P(n')$ holds for all $n' < n$. Our goal is to prove that $P(n)$ holds. We consider three cases according to $n \pmod{3}$:

Case (i) $[n \equiv 0 \pmod{3}]$: Then n is a multiple of 3, so Anna immediately writes $n/3$ on the board. Since $n/3 < n$, we know that $P(n/3)$ holds, and therefore Anna eventually writes a 1.

Case (ii) $[n \equiv 1 \pmod{3}]$: In this case Anna first writes $n + 1$, then $n + 2$, then $(n + 2)/3$. Since $(n + 2)/3 < n$ we know that $P((n + 2)/3)$ holds, so Anna eventually writes a 1.

Case (iii) $[n \equiv 2 \pmod{3}]$: Here Anna writes $n + 1$, then $(n + 1)/3$ and the argument proceeds as before.

Thus in all three cases Anna eventually writes a 1 on the board, so $P(n)$ holds and we are done.

4 Intensive Swim Workout

A swimmer swims in a pool where 70 laps is equivalent to 1 mile. (By “lap” here we mean one length of the pool.) She uses the following exhausting workout regime:

- In alternate laps, she puts her fins on and takes them off.
- She cycles through five breathing patterns (labeled 1,2,3,4,5), switching to the next one at the end of each lap.
- She swims seven different strokes (1=freestyle, 2=breaststroke, 3=butterfly, 4=backstroke, 5=front kick, 6=back kick, 7=sidestroke), switching to the next one at the end of each lap.

She always starts her workout with fins off, breathing pattern 1, freestyle.

One day she loses count of how many laps she swam, but she does know she swam more than one mile and less than two miles. She also knows that her last lap was with fins on, breathing pattern 3, and back kick.

- Explain clearly why she has enough information to deduce exactly how many laps she swam.
- How many laps did she swim? [**Note:** You may find it easier to do this calculation by hand rather than using a standard algorithm.]

Solution:

- Let L denote the total number of laps swum. We are given that $70 < L < 140$, so if we can compute $x = L \bmod 70$ we will be done. From the information about the initial and final laps, we know that

$$\begin{aligned}x &\equiv 0 \pmod{2}; \\x &\equiv 3 \pmod{5}; \\x &\equiv 6 \pmod{7}.\end{aligned}$$

By the Chinese Remainder Theorem, since 2, 5, and 7 are coprime and their product is 70, these three congruences have a *unique* solution $x \pmod{70}$. (Note: It is important that the solution is unique; otherwise, we would not be able to compute L with certainty.) We can then deduce that $L = 70 + x$.

118. As explained in part (a), the answer is $70 + x$, where x is the unique solution $\pmod{70}$ to the three congruences above. We could use the CRT algorithm in the notes to solve these, but the numbers are small enough that we can more quickly do this by hand. In particular, the only numbers $\pmod{70}$ that are simultaneously congruent to 6 $\pmod{7}$ and to 3 $\pmod{5}$ are 13 and 48, and the only one of these that is congruent to 0 $\pmod{2}$ is $x = 48$.

5 Graph Proofs

All graphs are simple and undirected unless otherwise specified.

- Consider a simple connected planar graph with e edges and $v > 2$ vertices. Prove that if $e < 3v - 6$, then there is a face of size at least 4 (that is, there exists a face bounded by at least 4 edges). (Recall that in a simple planar graph, every face has size at least 3.)
- Consider a bipartite graph $G = (V, E)$ with an Eulerian tour. Give a method to partition the edges of the graph into two graphs where every vertex has exactly half the degree, and justify your answer.

That is, form $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, where $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$, and every vertex v has $d_1(v) = d_2(v) = d(v)/2$ where $d(v)$ is the degree of a vertex in G , and $d_i(v)$ is the degree of v in G_i .

Solution:

- (a) Suppose for contradiction that every face has size exactly 3 (the same argument can be used in a proof by contraposition). This means that $\sum_{i=1}^f s_i = 3f = 2e$ by counting face-edge adjacencies. Using Euler's formula, we then have $v + \frac{2}{3}e = e + 2$ or that $e = 3v - 6$. However, we know that $e < 3v - 6$, which is a contradiction; this means that there must be a face of size strictly larger than 3, i.e., there is a face of size at least 4.

Alternatively, one can use a direct proof. By Euler's formula, $v + f = e + 2$, so $v = e - f + 2$. Plugging into the inequality, we have that $e < 3v - 6 = 3(e - f + 2) - 6$, which rearranges to form $3f < 2e$. We also know by face-edge adjacencies that $\sum_{i=1}^f s_i = 2e > 3f$, so at least one of the s_i 's must be strictly larger than 3, i.e., there must be some face of size at least 4.

- (b) Traverse the tour, and place the odd numbered edges in E_1 and the even ones in E_2 .

Since the graph is bipartite, the tour will enter one side in odd steps, and the other in even steps, so E_1 is the entering edges for one side (and leaving for the other), and E_2 is the leaving for one side (and entering for the other side). The number of entering and leaving edges for each vertex is the same, thus each vertex has half the degree in G_1 and G_2 .

Note that the bipartite condition must be used since the algorithm fails on a triangle, which is Eulerian.

6 Sleepiness

- (a) Consider a set of intervals $[s_1, e_1], \dots, [s_n, e_n]$ where s_i is the start time and e_i is the end time of an interval. The associated interval graph has a vertex for each interval and an edge between any pair of intervals that overlap; for example, if $i_1 = [3, 5]$ and $i_2 = [4, 6]$ and $i_3 = [6, 7]$, we have edges $\{i_1, i_2\}$ and $\{i_2, i_3\}$ but no edge between i_1 and i_3 .

Prove that if there is a cycle in an interval graph then there is a point in time where at least 3 intervals overlap.

- (b) Seven students each fall asleep three times during CS 70 lecture. Furthermore, for every two of these seven students, there exists a time when both of them are sleeping. Prove that there must be some time when at least three students were sleeping at once.

Solution:

- (a) Suppose for contradiction that no point in time lies in three intervals. Let the cycle be $n_1, n_2, \dots, n_k$, labeled so that n_1 has the smallest start time, and let t_i be the start time of n_i . We use one fact: if two intervals overlap, the later of their start times lies in both.

First, the start times strictly increase along the cycle: $t_2 < t_3 < \dots < t_k$. Suppose $t_{i-1} \leq t_i$ for some $2 \leq i < k$; this holds for $i = 2$ by the choice of n_1 . Since n_{i-1} and n_i overlap, t_i lies in both. If $t_{i+1} \leq t_i$, then t_i would also lie in n_{i+1} , a third interval. So $t_{i+1} > t_i$.

Now n_k overlaps n_1 and $t_1 \leq t_k$, so t_k lies in n_1 . It also lies in n_k , and in n_{k-1} , since n_{k-1} overlaps n_k and $t_{k-1} < t_k$. That is three intervals overlapping at time t_k , a contradiction.

- (b) Give each student three vertices, representing each of the times they fell asleep. Draw an edge between two vertices if the respective naps overlap. There are $7 \cdot 3 = 21$ vertices and at least $\frac{7 \cdot 6}{2} = 21$ edges, one for each pair of students. This graph must have a cycle: if it were acyclic, each connected component would be a tree with one fewer edge than vertices, so it would have at most $21 - 1 = 20$ edges. By part (a), there must be three naps that overlap.