Cooperative Apartment

A

We will set all people as $\{p_1,\ldots,p_n\}$ and the nights as $\{d_1,\ldots,d_n\}$. To build a bipartite graph, we will set all people as vertices $P = \{p_1,\ldots,p_n\}$ and nights as $D = \{d_1,\ldots,d_n\}$. We will state that a perfect matching in G exists if every single person is assigned a different night to cook with no overlaps. Likewise, if every single person is assigned a different night to cook with no overlaps, a perfect matching in G exists. Here, we will set the matching of each persons to a night represented by an edge in E, where GE = $\{(p_i, d_j)\ldots(p_n, d_n)\}$. (p_i, d_j) will represent that person i is available to cook on day j given that $d_j \notin \_S_i$ where S_i represents all days where person p_i is unable to cook. Within graph G, each person P will be matched with each day D, where $|P| = |D|$ would hold true.

B

A feasible dinner schedule is defined as an assignment of each person to a different night where each person cooks on exactly one night. Alanis has already created a feasible schedule for n-2 people. We want to fix this schedule in $O(n^2)$ time. A valid schedule would be a bijection between the set of people P and the set of days D. Since p_i and p_j are both assigned to cook dinner on the same night d_k, and no one to cook on d_l, Alanis will technically only have to un-match one person from p_i or p_j and rematch that person to another day d_l. Using the disconnected person, we will use that person and determine a new edge between (p_i, d_j) and update the set E which contains n-2 correct matchings. If such path exists, the FIX_DINNER_MATCHINGS algorithm will output the correctly updated dinner schedule.

Algorithm FIX_DINNER_MATCHING(S_i, S_j G.P, G.D)

// we will assume Alanis properly matched n-2 edges in graph G
// a nested for-loop will be used to add each matching edge (p_i,d_j) to set E
// set E will contain a supposed "bijection" where n-2 edges of (p_i,d_j) are correctly matched

for each vertex $p_i \in \_G.P$ // for each person p_i in P
for each vertex $d \in \_G.D$ // nested for-loop $O(n^2)$
if p_i is connected to d_j // if edge exists between p_i and d_j
add each edge to set E // E = $\{(p_i, d_j),\ldots,(p_n, d_n)\}$

if dk ∉ Si and dl ∉ Sj
update to (p_i, d_k) and (p_j, d_l) in set E // E will now contain the correct dinner schedule
for each edge $e \in \_E$ // iterate over each edge O(n) time
print("Updated dinner matchings:" + E) // output correct dinner schedule
break
else if $d_k \notin \_S_i$ **and** $d_l \in \_S_j$
print("No feasible schedule exists. P_j does not have any available days on d_l")
break
else if $d_l \notin \_S_i$ **and** $d_k \notin \_S_j$
update to (p_i, d_l) and (p_j, d_k) in set E
for each edge $e \in \_E$. // iterate over each edge O(n) time
print("Updated dinner matchings:" + E)

```
break
else
print("No feasible schedule exists.")
```

CLRS page 1167 was utilized for definitions on bijections – a one-to-one correspondence where each element in X is matched with exactly one element in Y. Hence |X|=|Y|. If a perfect schedule exists, this would hold true for above where |P|=|D|.