Viraj Agashe
2020CS10567
cs1200567@cse.iitd.ac.in

CON101: Assignment 4a
P-NP Problems

2021-09-25

1 Problem 1

We begin by noting that the number of edges (say n) which are present in the maximum matching satisfy $n \leq 4$. Now we have,

- For G_1 the maximum matching is $\{(3,7),(4,8),(2,5),(1,6)\}$.
- For G_2 the maximum matching is $\{(3,7),(4,8),(2,5),(1,6)\}$.

2 Problem 2

By definition, a matching with $|E|$ edges is maximum if for any other matching with $|E_i|$ edges, $|E_i| \leq |E|$. The maximum matching need not be unique. A maximum matching can be found in both the graphs 3 and 4, which has more number of edges to the given matchings.

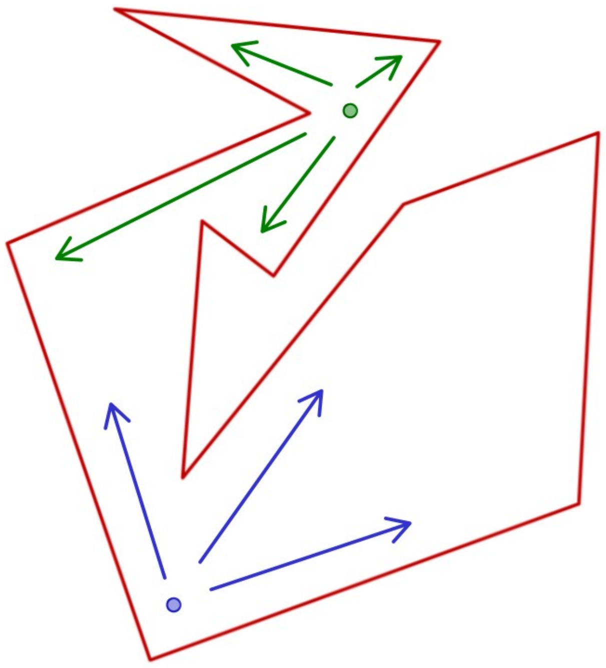
2.1 Graph 3

Consider $\{(w_1,v_1),(w_2,v_2),(w_3,v_4),(w_4,v_3)\}$, which is a matching with four edges which is greater than the matching given in the question. Hence it is not the maximum matching.

2.2 Graph 4

Consider $\{(w_1,v_7),(w_2,v_1),(w_3,v_3),(w_5,v_2),(w_6,v_6),(w_7,v_8)\}$ is a matching with six edges which is greater than the matching given in the question, which has 5 edges. Hence it is not the maximum matching.

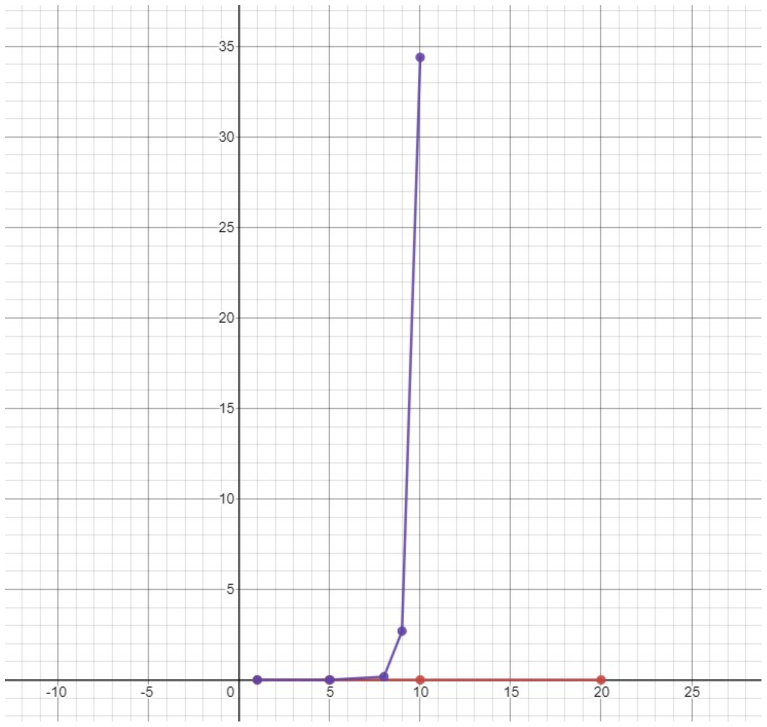
3 Problem 3



We claim that the minimum number of guards required for this problem is 2. The guards will be placed at the positions shown in the figure. Clearly they can survey the entire gallery. We argue that 1 guard will not be sufficient for this. When there is only one guard, the problem is, if we want a guard to be able to see the topmost triangular portion, he/she must be at the position occupied by green. But clearly, green cannot see any of the lower right portion of the gallery. So minimum 2 guards are required.

4 Problem 4

I got the following plot of the times taken by permutation sort and insertion sort. Permutation sort was taking too much time for $n > 10$ so further values have not been plotted.



I used the following program to measure the times taken by permutation sort.

```python
import time

def perm(l):
    if len(l) == 0:
      return []
    elif len(l) == 1:
      return [l]
    else:
      temp = []
      for i in range(len(l)):
          n = l[i]
          left = l[:i] + l[i+1:]
          for p in perm(left):
              temp.append([n] + p)
      return temp

begin = time.time()

for p in perm(ls):
  j = 1
  for i in range(len(p)-1):
    if p[i] < p[i+1]:
      j+=1
  if j == len(p):
    print(j)
    end = time.time()
    print(end-begin)
    break
```