CSE 60111: Complexity and Algorithms
Homework 8

---

1. Consider two sets A and B each having n integers in a range from 0 to $10n$. We wish to compute the cartesian sum of A and B, defined as $C = \{x + y \mid x \in A, y \in B\}$. Compute the number of elements of C and the number of times each element of C is realized as a sum of elements from A and B, in $O(n \log n)$ time.

2. One of your fellow students, just after class, suggests the following greedy variant to the generic Ford-Fulkerson augmenting path algorithm. Instead of maintaining the residual graph, just reduce capacity along edges of the augmenting path. In particular, if you saturate an edge e (so that $f(e) = c(e)$), remove the edge from the graph. Who needs all that residual graph messiness, anyway? This gives the following algorithm:

```
GreedyFlow(G, c, s, t):
  for every edge e in G
      f(e) ← 0
  while there is a path from s to t in G
      π ← an arbitrary path from s to t
      F ← minimum capacity of any edge in pi
      for every edge e in π
          f(e) ← f(e) + F
          if c(e) = F
              remove e from G
          else
              c(e) ← c(e) − F
  return f
```

   Of course, we have already seen that this algorithm won't necessarily return a maximum flow, since bad choices of paths can get you stuck. Show that GreedyFlow is not even guaranteed to compute a good approximation to the maximum flow. That is, for any constant $\alpha > 1$ there is a flow network G such that the value of the maximum flow is more than α times the value of the flow computed by GreedyFlow.

3. A cycle cover of a given directed graph G = (V, E) is a set of vertex-disjoint cycles that cover every vertex in G. Describe and analyze an efficient algorithm to find a cycle cover for a given graph, or correctly report that no cycle cover exists.