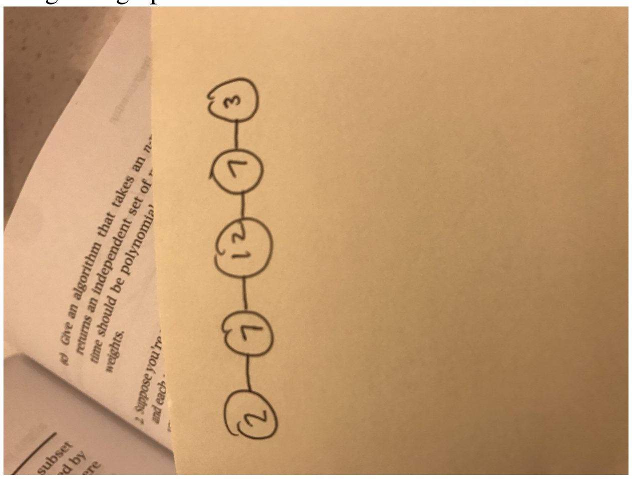
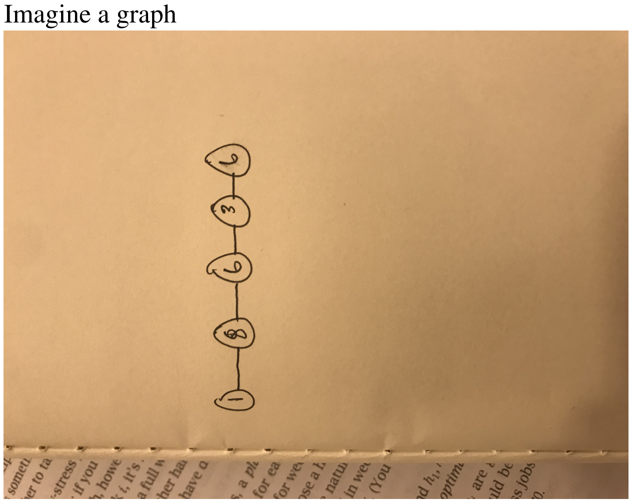
a) Imagine a graph




If we pick the heaviest first, then we get 12 as our answer, but the optimal solution is 14

b) Imagine a graph




If we used this algorithm, then we would pick the odd set and get 13 as our answer, but the optimal solution is 14.

c) Solution:
We can imagine a solution to this problem that mirrors the solution of the Weighted Interval Problem.

We have $n$ nodes in a graph $G$ and we want to find the heaviest independent set.

We can number these nodes $1...n$. We can define $p(i)$ to be the heaviest independent set of nodes $1...i$ such that $1 \leq i \leq n$.

Consider an optimal solution $O$. For a set of $n$ nodes, this solution maximizes $p(n)$.

This solution contains node $n$ or it does not. If $n \in O$ then the optimal solution is $p(n)$. Otherwise, if $n \notin O$, the optimal solution is $p(n-1)$.

This leads us to the fact that the optimal solution $p(n) = max(p(n), p(n-1))$

We can then compute a memoized version of this recursive function:

```
Maintain an array M[0...n]

Set M[0] = 0

For all nodes $i$
      Compute the value of p(i) and store it at M[i].

 To find the optimal solution for n nodes, compute max(M[n], M[n - 1]).
```