CS 4820, Spring 2019 **Homework 3, Problem 1**
Name: Kevin Klaben NetID: kek228 Collaborators:

(1) *(10 points)* In American politics, *gerrymandering* refers to the process of subdividing a region into electoral districts to maximize a particular political party's advantage. This problem explores a simplified model of gerrymandering in which the region is modeled as one-dimensional. Assume that there are $n > 0$ *precincts* represented by the vertices $v_1, v_2, \dots, v_n$ of an undirected path. A *district* is defined to be a contiguous interval of precincts; in other words a district is specified by its endpoints $i \le j$, and it consists of precincts $v_i, v_{i+1}, v_{i+2}, \dots, v_j$. We will refer to such a district as $[i, j]$.

Assume there are two parties A and B competing in the election, and for every $1 \le i \le j \le n$, $P[i, j]$ denotes the probability that A wins in the district $[i, j]$. The probability matrix P is given as part of the input. Assume that the law requires the precincts to be partitioned into exactly k disjoint districts, each containing at least $s_{\min}$ and at most $s_{\max}$ nodes. You may also assume the parameters $n, k, s_{\min}, s_{\max}$ are chosen such that there is at least one way to partition the precincts into k districts meeting the specified size constraints. Your task is to find an efficient algorithm to gerrymander the precincts into k districts satisfying the size constraints, so as to maximize the expected number of districts that A wins.

```
RECGERRY(k,i)
if T[k][i]!=NULL then
   return T[k][i]
else if k=1 then
   if (n-i>Smax or n-i<Smin) then
      T[k][i]=0
      return T[k][i]
   else
      T[k][i]=p[i,n]
      return p[i,n]
   end if
else
   if i>n then
      return 0
   end if
   maxl=[]
   maxj=-1
   maxp=0

   for j in range(Smin, min(Smax,n) do
         (m,l)=RECGERRY(i+j+1,k-1)
         if m+P[i,i+j]>maxp then
         maxP=m+P[i,i+j]
         maxl= l+[j]
      end if
   end for
   t[k][i]=(maxp,maxl)
```

```
    return t[k][i]
  end if
```

Runtime Analysis This algorithm memoizes the running time by the 2 dimensional array T. As T is k by i and takes a maximum of Smax-Smin executions of the for loop for each value put into T, the time for memoizing is O(ki(Smax-Smin)). Overall, there are possibly Smin-Smax calls to RECGERRY in each time that RECGERRY is called, and RECGERRY can be called k times as k decreases. Thus the overall runtime is O(ki(Smax-Smin)+(Smax-Smin)^{k}).

Proof of Correctness. Claim: This algorithm achieves the optimal chance of winning the highest number of districts. Proof: For sake of contradiction suppose that there exists some set of districts satisfying the size conditions that provides a higher probability of winning more districts than that which the algorithm output. We know that both sets contain the same number of districts k and thus that in this case at least one of the districts in in this set has a higher probability of being won than at least one of the corresponding districts in the algorithm output. This district and all others past it make up some number x¡=k districts that satisfy the size constraints. Let this higher probability district begin at i1 and end at j1. Thus when the algorithm was performed for i=i1 and k=x, this set of districts would have been output and selected as it is optimal. However, this set was not and therefore does not exist as its probability would have been higher and in turn selected. Thus, the algorithm outputs the correct optimal solution.