USACO OCT08 Problem 'quad' Analysis

by Richard Peng

The first observation to make is that 4 edges can make a polygon of perimeter N iff their lengths are all in the range 1..n/2. Then the problem becomes counting the number of ways 4 integers in the range 1...k sums to N, which can be done with dynamic programming.

The DP state is num[i,s], which represents the number of ways of making a sum of s using i such numbers. Then the base case is when i=0, where num[0,0]=1 and num[0,s']=0.

Now for the transition, suppose the ith number of x, then we have num[i-1,s'-x] contributing to num[i,s']. So the transition is num[i,s']=sum(1<=x<=min(s',[n/2]),num[i-1,s'-x]. Computing this naively using a for loop gives an O(N^2) algorithm, and any type of optimization by precomputing partial sums or two pointer walks would give a O(N) algorithm.

Below is Jaehyun Park's solution:

```
#include<stdio.h>
long long d[3000];
int main() {
    int n;
    FILE *ifp=fopen("quad.in", "r"), *ofp=fopen("quad.out", "w");
    d[5]=4;
    for(n=7;n<=2500;n+=2) d[n]=d[n-2]*(n+1)/(n-5);
    for(n=6;n<=2500;n+=2) d[n]=d[n-1]+(n-1)/2;
    fscanf(ifp, "%d", &n);
    fprintf(ofp, "%lld\n", d[n]);
    fclose(ifp);
    fclose(ofp);
    return 0;
}
```