CS124 Programming Assignment 3
Andrew Paek & Jason Shen

Give a dynamic programming solution to the Number Partition problem.

Given a set of integers S that we would like to optimally partition, the loose recurrence relation for our dynamic programming solution is the following:

$$f(A + \{x\},\ i) = f(A) \quad || \quad f(A,\ i - x)$$

where A is a subset of S, x is an element in S not in A, and $i \in [0,\ b]$ where b is the sum of all numbers in S. The second part of the disjunction is only considered for $i - x \geq 0$. The function $f(A,\ i)$ returns True if some combination of numbers in A sum to i and False otherwise.

This recurrence relation has the base case:

$$f(B,\ 0) = True$$
$$f(B,\ i) = False \text{ for } i > 0$$

where B is the empty set $\{\}$.

In order to truly make this solution an example of dynamic programming, we store the results of each $f(A,\ i)$ in a large matrix of size n by b where $n = |A|$ and b is the total sum of numbers in S. The columns of the matrix will be in the sets of each step, starting with the empty set and adding one member at a time until it becomes the full set, and the rows are the values of i from 0 to b. This is illustrated in the simple example table below:

For an input set $S = \{3,\ 1,\ 4,\ 7,\ 5\}$

	{}	{3}	{3, 1}	{3, 1, 4}	{3, 1, 4, 7}	{3, 1, 4, 7, 9}
0	T	T	T	T	T	T
1	F	F	T	T	T	T
2	F	F	F	F	F	F
3	F	T	T	T	T	T
4	F	F	T	T	T	T

5	F	F	F	T	T	T
6	F	F	F	F	F	F
7	F	F	F	T	T	T
8	F	F	F	T	T	T
9	F	F	F	F	F	T
10	F	F	F	F	T	T
11	F	F	F	F	T	T
12	F	F	F	F	T	F
13	F	F	F	F	F	T
14	F	F	F	F	T	T
15	F	F	F	F	T	T
16	F	F	F	F	F	T
17	F	F	F	F	F	T
18	F	F	F	F	F	F
19	F	F	F	F	F	T
20	F	F	F	F	F	T

The time complexity of this algorithm is $O(bn)$ since we are creating a table of size bn. Once we have created the table we would simply be able to find the smallest set such that its members sum to either $b/2$ or the closest number to $b/2$ either above or below. When we find such a set we can say that that set is the first set of our partition and the remaining numbers in S become members of our other set. By looking for the closest number to $b/2$ as the sum of one of the two sets, we know that we are getting the smallest possible residue.

Explain briefly how the Karmarkar-Karp algorithm can be implemented in *O(n log n)* steps, assuming the values in A are small enough that arithmetic operations take one step.

CS124 Programming Assignment 3
Andrew Paek & Jason Shen

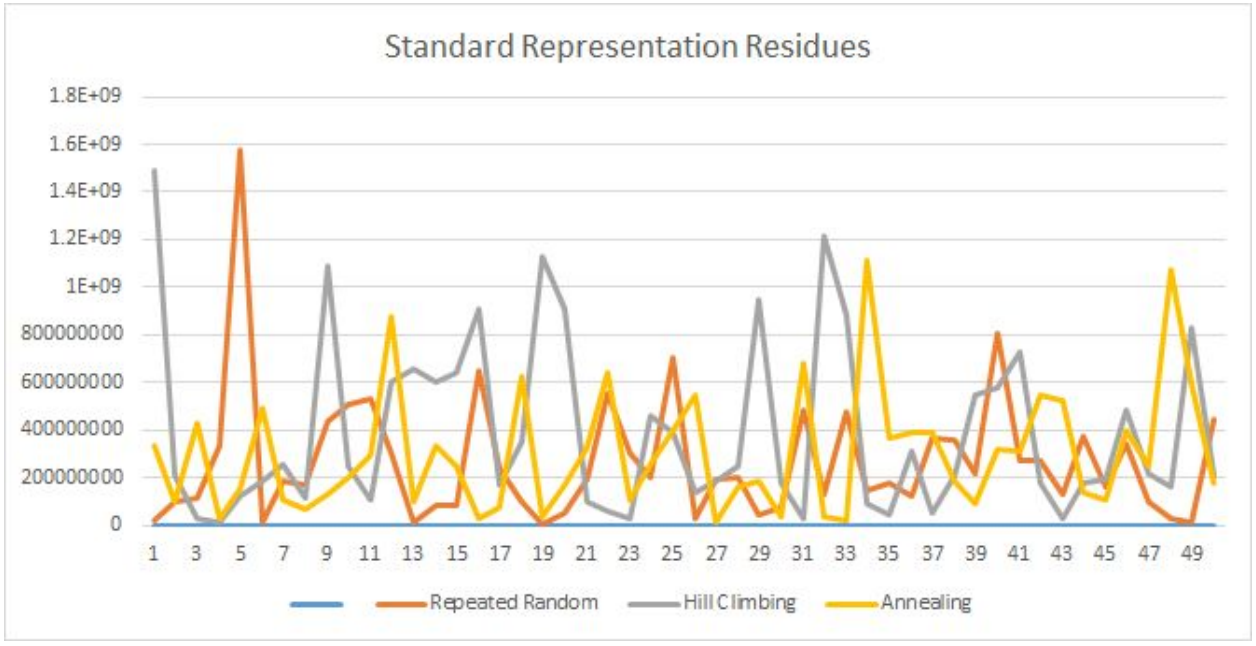
The Kamarkar-Karp algorithm turns one member of the input set S into 0 during each step. Therefore, we will require $n-1$ steps in order to arrive at a residue. During each step we would like to find the two largest values in time linear in $log\ n$. In order to do this we will throw the entire input set S into a max-heap. The time to obtain the two largest values and rebalance will be linear in $log\ n$ since the time to obtain the max is constant while the time to rebalance is $log\ n$. Therefore, we will have n steps in our Karmarkar-Karp algorithm with each step having a time complexity of $k * log\ n$. Therefore, the overall time complexity of our algorithm will be $O(nlog\ n)$.

Discussion of Results:

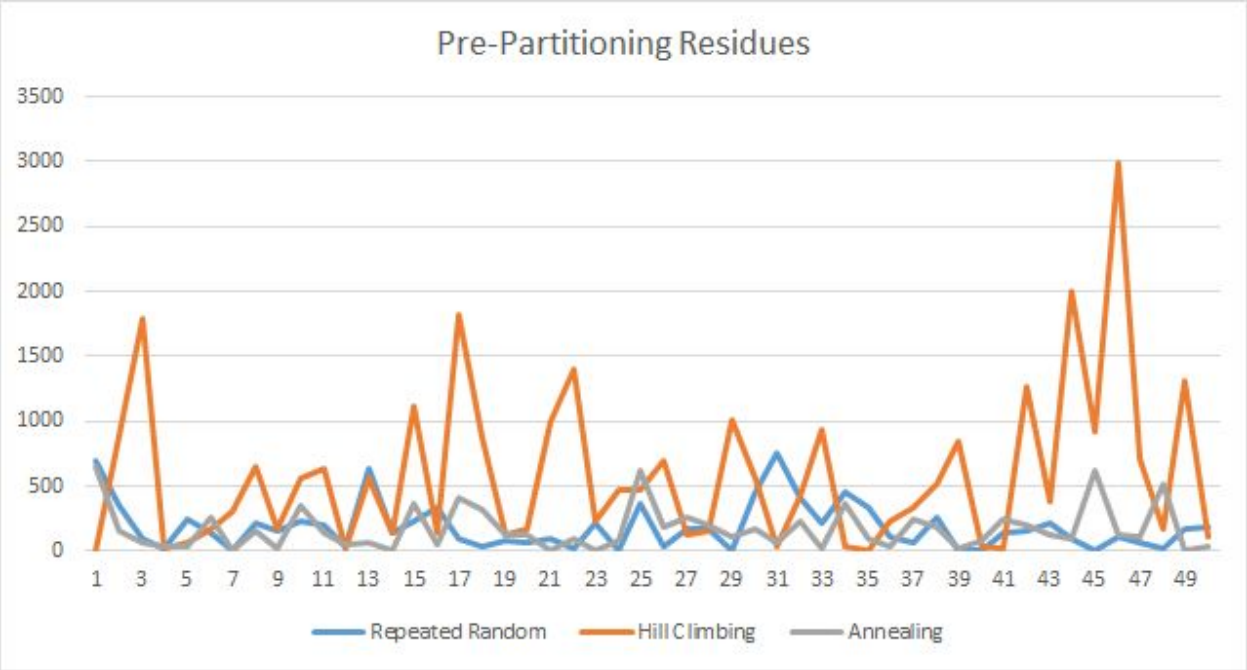
The best combination of algorithm and representation that provided the best average residual was the Annealing algorithm with the Pre-Partitioning representation. This combination returned an average residual of 168.98, lower than all the other combinations.

The table of all our results is attached in pa3table.pdf. But below are the averages and the graphs of our results.

Karmarkar-Karp Residuals and Runtimes		Standard Representation Residuals and Runtimes						Prepartitioning Representation Residuals and Runtimes					
[redacted]		Repeated Random		Hill Climbing		Annealing		Repeated Random		Hill Climbing		Annealing	
284894.44	0.02	269338337.6	35.84	395637160	4.66	304489380.6	13.38	185.16	315.4	591.78	205.24	168.98	223.56



CS124 Programming Assignment 3
Andrew Paek & Jason Shen



It is clear that pre-partitioning gives us the best results for finding the optimal partition. Pre-partitioning outperforms the standard representation by several orders of magnitude for all three algorithms. However, pre-partitioning does seem to take a bit longer to run than the standard representation. The Karmarkar-Karp algorithm takes the shortest time to run, but does not return anywhere close to best residuals.

CS124 Programming Assignment 3
Andrew Paek & Jason Shen

Discuss briefly how you could use the solution from the Karmarkar-Karp algorithm as a starting point for the randomized algorithms, and suggest what effect that might have.

We expect this procedure to improve our residuals from the normal Karmarkar-Karp approach. Starting out with a better partition will essentially allow us to run more iterations to improve a superior residue and save time in the algorithm. Therefore, we believe it will improve the performance of the algorithm in terms of the final residue it produces.