Number Partition Problem

Eliel Dushime

April 2022

1 Introduction

For this assignment, we implement a number of heuristics for solving an NP-complete problem, the Number Partition problem. We make use of two different representations (standard representation and a prepartitioning representation).

Our aim will entail comparing results from a variety of algorithms (the Karmarkark-Karp algorithm and a variety of heuristic algorithms applied to random input sets and making use of each of the mentioned representations: the Repeated Random, Hill Climbing, and Simulate Annealing heuristics).

The entire code is found in partition.py, with inputfile as a text file that contains a list of 100 (unsorted) integers, one per line. The output is the residue of this list of 100 integers.

The submitted code is executed with the run command:

./partition flag algorithm inputfile

2 DP solution to the Number Partition problem

Given a sequences of n numbers $A = a_1, a_2, \ldots a_n$, the Number Partition problem is to decide whether A can be divided into two subsets A_1 and A_2 of equal sums.

We solve the problem dynamically as follows:

Pseudocode:

1. Calculate the Sum of numbers in given sequence of numbers. If Sum is odd then return 0.

2. We create a 2D-array D of size $Sum/2 + 1$ by $n + 1$, where sum is the sum of all the numbers in the sequence. The row of D represents a sum while its column represents the size of an array. The entries in the 2D-array will be bits

(0 or 1). The value of $D[i][j]$ will be 1 if there exists a subset of size less than or equal to j and sum equal to i.

3. Populate the 2D-array in a bottom-up fashion as follows:

   - Fill in the base cases: $D[0][i] = 1$ and $D[i][0] = 0$ if $i < Sum$
   - Recursively build the rest of the array from $i = j = 1$ with $D[i][j] = 1$ if $D[i][j-1] = 1$ or $D\Big[i - A[j-1]\Big][j-1] = 1$ (for $i \geq A[j-1]$)

4. We return $D[Sum/2][n]$

This algorithm takes $n * b$ with b the Sum

3 Karmarkar-Karp implementation

To implement the Karmarkar algorithm, we will need a way to extract the two largest numbers from the input sequence at each step of the algorithm. A max heap therefore is a good strategy.

Applying max heapify to get the two largest numbers from the heap will take $log(n)$. Since there will be at most n steps to implement differencing, we get overall run time of $O(nlog(n))$.

4 Experiments

Country List			
Code	Algorithm	Mean	Median
0	Karmarkar-Karp		
1	Repeated Random		
2	Hill Climbing		
3	Simulated Annealing		
11	Prepartitioned Repeated Random		
12	Prepartitioned Hill Climbing		
13	Prepartitioned Simulated Annealing		

It is clear that the pre-partition representation performs much better than the standard representations in all 3 randomized heuristic algorithms, but a significant factor. It is also better than the Karmarkar-Karp algorithm. Among the pre-partition representations, the Repeated Random performs best. It is followed respectively by Simulated Climbing and Hill Climbing. Both means and medians show this discrepancy.

5 Karmarkar-Karp algorithm as a starting point for the randomized algorithms

This Karmarkar-Karp algorithm gives a good approximation for the actual residue. One may use it as a starting point to optimize the other algorithms. We would constrain the starting random solution to have a residue less than that given by the Karmarkar-Karp algorithm.This allow better searches of the state space and avoid bad neighbors.