

Subset Sum

Advanced Algorithms (CSE 794)

Instructor: Tamal K. Dey

Scribe: Andrew G Slatton *

May 28, 2009

1 Introduction

Given a set of integers, S , and a single integer, w , the subset problem (a special case of the knapsack problem) is to find $S' \subseteq S$, such that $\sum_{x \in S'} x = w$. There are two versions of this problem: the decision version and the optimization version. In the decision version, one searches for the existence of S' such that $\sum_{x \in S'} x = w$. The optimization version searches for S' such that the sum of all elements in S' is maximal provided that $\sum_{x \in S'} x \leq w$. Both of these versions are NP-complete. Herein we discuss an exponential-time algorithm for finding the exact solution to the subset problem, a fully-polynomial time approximation algorithm for solving subset sum, and a dynamic programming algorithm for solving the knapsack problem.

2 Preliminaries

We set up some basic notations and concepts that are needed to describe the subset sum algorithms.

2.1 Notations

Given a set $S = \{x_1, x_2, \dots, x_i\}$, we must often compute the sum of some integer, y , with each of the members of S and return the resulting set. We will define $S+y = \{x + y | x \in S\}$ to represent this operation.

2.2 Merging Lists

The merging of two sorted lists into a single sorted list is a well-known concept in computer science, most commonly associated with the MERGESORT algorithm. We will call the function for performing this task $\text{MERGELISTS}(L_1, L_2)$. This function takes linear time with respect to the lengths of the lists. Further information can be found at [1].

*Department of Computer and Information Sciences, The Ohio State University, Columbus, Ohio, USA.

3 Exact Algorithm

3.1 Algorithm

```
EXACTSUBSETSUM( $S, w$ )  
  
1.  $n := |S|$ ;  
2.  $L_0 := \{0\}$ ;  
3. for  $i := 1$  to  $n$   
    $L_i := MergeLists(L_{i-1}, L_{i-1} + x_i)$ ;  
   remove elements from  $L_i$  which are greater than  $w$ ;  
4. endfor  
5. return the largest element in  $L_n$ .
```

Here, S is the set upon which the subset sum is being computed, w is the target value, and x_i is the i^{th} element of S . Note that this algorithm only works when $x_i \geq 0 \forall i$. In order to make it work for S containing some $x_i < 0$, we must omit the line that removes elements greater than w from L_i .

3.2 Analysis

This algorithm runs in $O(2^n n)$ time. The analysis is fairly trivial: in the worst case, L_{i-1} and $L_{i-1} + x_i$ share no elements and so L_i is twice the length of L_{i-1} ; the for-loop containing MERGELISTS runs n times, so we have $\sum_{i=1}^n 2^i = O(2^n n)$.

4 Polynomial Time Approximation

4.1 Algorithm

This algorithm relies on trimming the list of subset sums, L by a parameter ε , removing elements from L in a way such that if $L' = \text{TRIM}(L, \varepsilon)$ then for each y removed from L , $\exists z \in L'$ such that $\frac{y}{1+\varepsilon} \leq z \leq y$.

```
APPROXSUBSETSUM( $S, w, \varepsilon$ )  
  
1.  $n := |S|$ ;  
2.  $L_0 := \{0\}$ ;  
3. for  $i := 1$  to  $n$   
    $L_i := MergeLists(L_{i-1}, L_{i-1} + x_i)$ ;  
    $L_i := Trim(L_i, \frac{\varepsilon}{2^n})$ ;  
   remove elements from  $L_i$  which are greater than  $w$ ;  
4. endfor  
5. return the largest value in  $L_n$ ;
```

Here, ε is the approximation variable and the TRIM function is defined below.

```

TRIM( $L, \varepsilon$ )
1.  $m := |L|$ ;
2.  $L' := \{y_1\}$ ;
3.  $last := y_1$ ;
4. for  $i := 1$  to  $n$ 
    if  $y_i > last(1 + \varepsilon)$  then
        append  $y_i$  to the end of  $L'$ ;
         $last := y_i$ ;
    endif
5. endfor
6. return  $L'$ ;

```

In this function, y_i is the i^{th} element of L . It follows that $last$ is the element from L most recently appended to L' .

4.2 Analysis

Theorem 1. APPROXSUBSETSUM is a fully-polynomial time scheme for the subset sum problem.

Proof. Let $y^* \in P_n$ be the optimal solution and let z^* be the output of the algorithm.

$$z^* \leq y^*$$

For each $y \in P_i, \exists z \in L_i$ such that $\frac{y}{(1+\frac{\varepsilon}{2n})^i} \leq z \leq y$.

In particular, we have $\frac{y^*}{(1+\frac{\varepsilon}{2n})^n} \leq z \leq y^*$.

$$\implies \frac{y^*}{z^*} \leq \left(1 + \frac{\varepsilon}{2n}\right)^n$$

$$\lim_{n \rightarrow \infty} \left(1 + \frac{\varepsilon}{2n}\right)^n = e^{\frac{\varepsilon}{2}}$$

so the approximation factor is $\left(1 + \frac{\varepsilon}{2n}\right)^n = e^{\frac{\varepsilon}{2}} \leq 1 + \frac{\varepsilon}{2} + \left(\frac{\varepsilon}{2}\right)^2 \leq 1 + \varepsilon$

Consecutive elements in L_i differ at most by a factor of $\left(1 + \frac{\varepsilon}{2n}\right)$. Because the maximum value of any element in L_i is w , we know that $|L_n| \leq \log_{1+\frac{\varepsilon}{2n}} w = \frac{\ln w}{\ln(1+\frac{\varepsilon}{2n})} \leq \frac{2n(1+\frac{\varepsilon}{2n}) \ln w}{\varepsilon} \leq \frac{4n \ln w}{\varepsilon}$.

TRIM runs in $O(m)$ time, which we have determined to be equivalent to $O\left(\frac{n \ln w}{\varepsilon}\right)$, and MERGELISTS runs in $O(n)$ time, which is also $O\left(\frac{n \ln w}{\varepsilon}\right)$.

\therefore APPROXSUBSETSUM runs in $O\left(\frac{n^2 \ln w}{\varepsilon}\right)$ time. □

5 Dynamic Programming Approach

In this section, the function $opt(i, w) = \max_{S'} \sum_{x \in S'} x$ subject to $\sum_{x \in S'} x \leq w$ where $S' \subseteq \{x_1, x_2, \dots, x_i\} \subseteq S$.

DYNAMICPROGSUBSETSUM(S, w)

1. $n := |S|$;
2. initialize matrix M of size n -by- w ;
3. for $i := 1$ to n ;
 for $j := 1$ to w
 $M[i, j] = opt(i, j)$;
 endfor
4. endfor
5. return $M[n, w]$;

This algorithm runs in $O(nw)$.

References

- [1] Wikipedia - The Free Encyclopedia, Merge Algorithm