Example of Knapsack Algorithm

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on October 15, 2010 October 15, 2010 About Programming / Algorithms / ANSI/POSIX C

Example of Knapsack algorithm.

NOTIFICATION: These examples are provided for educational purposes. Using this code is under your own responsibility and risk. The code is given ‘as is’. I do not take responsibilities of how they are used.

knapsack.c:

```
/**
 * Knapsack Algorithm
 * @author: Alejandro G. Carlstein R. M.
 * @description: Simple Knapsack algorithm
 */
#include <stdio.h>

/* Macros Functions */
#define max(a,b) \
           ({ typeof (a) _a = (a); \
              typeof (b) _b = (b); \
            _a > _b ? _a : _b; })

/* Debugging flags */
#define DBG_LV1 0
#define DBG_LV2 0

/* Constants */
#define MAX_STRING_LEN 80
#define MAX_ITEMS 100
#define MAX_WEIGHT 1000

/* This is an item we can put into the knapsack. */
typedef struct{
  char name[MAX_STRING_LEN];
  int weight;
  int value;
} item;

/* Our dynamic programming table element.  We have the value
 * that we can store, and also a way to remember what we selected.
 */
typedef struct{
  int value;
  int prev_row;
  int prev_column;
```

```
} prog;

void print_table(int num_items, int max_weight);
void print_table_prev_xy(int num_items, int max_weight);

/* The actual DP table, and the items.  Note that we have an extra
 * row and column in the table for the 'zero elements' and 'no weight'
 * situation.
 */
prog table[MAX_ITEMS + 1][MAX_WEIGHT + 1];

item items[MAX_ITEMS];

int main(int argc, char *argv[]){

  int num_items; // Number of items
  int w; // Weight
  int max_weight;
  int n; // Item number;
  int i, j;
  int diff_weight;

  /* Read in the maximum weight and number of items. */
  scanf('%d %d', &max_weight, &num_items);

  if (DBG_LV2){
    printf('\nMax Weight: %d, Number of Items: %d \n\n', max_weight, num_items);
    printf('READ ITEMS\nw v name\n');
  }//end if

  /* Read in the items */
  for (n = 0; n < num_items; ++n){
      scanf('%d %d %s',
            &items[n].weight,
            &items[n].value,
            items[n].name);

     if (DBG_LV1)
        printf('%d %d %s\n',
               items[n].weight,
               items[n].value,
               items[n].name);
  }//end for

  if(DBG_LV2)
    printf('\n');

  /* Initialize the first row of the table. */
  for (w = 0; w <= max_weight; ++w){
      table[0][w].value = 0;
      table[0][w].prev_row = table[0][w].prev_column = -1;
  }//end for
```

```
// Fill in the table
// Your code goes here.
// Should be a for loop for the items, a for loop for the weight,
// and inside all of that, a few if statements to determine if you
// can take an item -- and if so, do you want to?
//
// I strongly recommend printing out EVERY DECISION your program
// makes while debugging things -- and feed your program very small
// problems until it's running.
//
// Debugging code is an important skill.  If you can work through a
// problem by hand, you should be able to get your code to solve the
// same thing.

/* Initialize the first column of the table */
for (i= 0; i <= num_items; ++i){
  table[i][0].value = 0;
}

if (DBG_LV2){
  printf('TABLE VALUES\n');
  print_table(num_items, max_weight);
}//end if

if (DBG_LV2){
  printf('TABLE PREVIOUS COORDINATES\n');
  print_table_prev_xy(num_items, max_weight);
}//end if

// Perform knapsack and find maximun value
for (i = 1; i <= num_items; ++i){

  for (w = 0; w <= max_weight; ++w){

  table[i][w].prev_row = i - 1;
  table[i][w].prev_column = w;

 // Check if item fit inside the knapsack

 if(items[i - 1].weight <= w) {

      int diff_weight = w - items[i - 1].weight;

      // Check which value is higher
  table[i][w].value = max((items[i - 1].value +
                        table[i - 1][diff_weight].value),
                        table[i - 1][w].value);

      // Keep track of the previous column
  if(table[i][w].value > table[i - 1][w].value)

   table[i][w].prev_column = diff_weight;
  }else{

   table[i][w].value = table[i - 1][w].value;

  }//end if

   }//end for

 }//end for

 if (DBG_LV2){
   printf('************\n\n');
 }//end if

 if (DBG_LV2){
   printf('TABLE VALUES\n');
   print_table(num_items, max_weight);
 }//end if

 if (DBG_LV2){
   printf('TABLE PREVIOUS COORDINATES\n');
   print_table_prev_xy(num_items, max_weight);
 }//end if

 // In my code, the maximum value is here.
 // I can use the prev_row and prev_column to trace back the solution.
 if (DBG_LV1)
   printf('Maximum value is %d\n', table[num_items][max_weight].value);

 if (DBG_LV2){
   printf('************\n\n');
 }//end if

 // Print results:

 w = max_weight;

 int count = -1;

int t;

int total_weight = 0;

 item *p_items[MAX_ITEMS];

for(i = num_items;
    i > 0;
    t = w,
    w = table[i][w].prev_column,
    i = table[i][t].prev_row){

 if(table[i][w].value != table[i - 1][w].value){
   total_weight += items[i - 1].weight;

   p_items[++count] = &items[i - 1];

   }//end if

 }//end for

 printf('%d\n', table[num_items][max_weight].value);

 printf('%d\n', total_weight);

  for (i = 0; i <= count; ++i){

    printf('%d %d %s \n',
           p_items[i]->weight,
           p_items[i]->value,
           p_items[i]->name);

  }//end for

}

/*
 *
 */
void print_table(int num_items, int max_weight){

  int w,i;

   for (i = 0; i <= num_items; ++i){
     for (w = 0; w <= max_weight; ++w){
       printf('%d ', table[i][w].value);
    }//end for
    printf('\n');
   }//end for
   printf('\n');

}

/*
 *
 */
void print_table_prev_xy(int num_items, int max_weight){

  int w,i;

   for (i = 0; i <= num_items; ++i){
     for (w = 0; w <= max_weight; ++w){
       printf('[%d,%d] ', table[i][w].prev_row, table[i][w].prev_column);
    }//end for
    printf('\n');
   }//end for
   printf('\n');

}
```

input.txt:

```
5
4
2 3 item_A
3 4 item_B
4 5 item_C
5 6 item_D
```

If you encounter any problems or errors, please let me know by providing an example of the code, input, output, and an explanation. Thanks.