Example of Huffman Algorithm by using Heap

acarlstein.com/

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

Example of Huffman algorithm by using heap.

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.

huffman.c:

```
/*
 * Program: Huffman using heap
 * Author: Alejandro G. Carlstein
 */

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX_CODE 1000000
#define MAX_LETTERS 4
#define MAX_HEAP 10
#define LEFT(i) ((i << 1) + 1)
#define RIGHT(i) ((i << 1) + 2)
#define PARENT(i) (((i + 1) >> 1) - 1)
#define DIV_BY_2(i) (i >> 1)
#define DBG_LV0 1
#define DBG_LV1 1
#define DBG_LV2 1
#define DBG_LV3 0
#define LETTER_A 65
#define DIR_LEFT '0'
#define DIR_RIGHT '1'

#define ROOT_INDEX 0

struct Data{
  int letter;
  int frequency;
  int left;
  int right;
  int parent;
}Data;

struct Data data[MAX_LETTERS];
int code[MAX_CODE][MAX_LETTERS];
int heap[MAX_HEAP];

void readInputLetters(void);
void readInputCode(void);
void printCodes(int *codes, int numCodes);
void printStructArray(struct Data array[], int length);
int huffman(void);
void buildMinHeap(int array[], int heapSize);
void minHeapify(int array[], int index, int heapSize);
int heapMin(int array[]);
int heapExtractMin(int array[], int *length);
void insertMinHeap(int array[], int value, int *length);
void exchange(int *a, int *b);
void printArray(int array[], int length);
int isLeft(struct Data array[], int indexLeft, int indexRight);
void test1();
void test2();

int num_letters;
int numCodes = 0;

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

 readInputLetters();

 readInputCode();

 if (DBG_LV3) printStructArray(data, num_letters);

 int i;
 for (i = 0; i < num_letters; ++i)
  heap[i] = i;

  printf('ROOT(huffman): %d \n', huffman());

 if (DBG_LV1) printStructArray(data, num_letters);

  //decodeMessage(data, num_letters, &code[0][0], numCodes);

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

 return 0;

}

void readInputLetters(void){
 if (DBG_LV0) printf('\nreadInputLetters()\n');

  for (num_letters = 0; num_letters < MAX_LETTERS; ++num_letters){
    scanf('%d', &data[num_letters].frequency);
    data[num_letters].letter = LETTER_A + num_letters;
    data[num_letters].left = -1;
    data[num_letters].right = -1;
  data[num_letters].parent = -1;

    heap[num_letters] = -1;
  }

}

void readInputCode(void){
 if (DBG_LV0) printf('\nreadInputCode()\n');

 int c;
  int indexCode = 0;

  numCodes = 0;

  while ((c = getchar()) != EOF){

    if (c != ' ' && c != '\n'){
      if (DBG_LV3) printf('[%d][%d]%c{%d}\n', numCodes, indexCode, c, c);
      code[numCodes][indexCode] = (int)c;
      ++indexCode;
    }

    if (c == '\n'){
      code[numCodes][indexCode] = -1;
      if (DBG_LV3) printf('{{%d}}\n', code[numCodes][indexCode]);
      ++numCodes;
      indexCode = 0;

    }
  }

 if (DBG_LV3){
   printf('CODES: \n');
   printCodes(&code[0][0], numCodes);

  }

}

void printCodes(int *codes, int numCodes){
  if (DBG_LV2) printf('printCodes(numCodes: %d)\n', numCodes);

  int indexCode, index;
 int indexArray;

  for (indexCode = 0; indexCode < numCodes; ++indexCode){

  indexArray = indexCode * sizeof(int);

    for (index = 0; codes[indexArray + index] > -1 ; ++index){

      printf('[%d][%d]: %c(%d) \n',
             indexArray, index,
             codes[indexArray + index], codes[indexArray + index]);

    }

    printf('\n');

  }

}

void printStructArray(struct Data array[], int length){
  if (DBG_LV0) printf('printStructArray()\n');

  int i;
  for (i = 0; i < length; ++i)
   printf('[%d]%c - %d (L:%d, R:%d, P:%d) \n',
          i, data[i].letter, data[i].frequency, data[i].left, data[i].right,
data[i].parent);
}

int huffman(void){
 if (DBG_LV0) printf('\nHUFFMAN()\n');

 int length = num_letters;
 int i;
 int left;
 int right;
 int parent;
 int n = length;

 length++;

 printf('length: %d\n\n', length);

 if (DBG_LV1) printArray(heap, num_letters);

 for (i = 0; i < n - 1; ++i){

  left = heapExtractMin(heap, &length);

  printf('length: %d\n\n', length);

  right = heapExtractMin(heap, &length);

  parent = left + right;

  data[left].parent = parent;
  data[right].parent = parent;

  printf('length: %d\n\n', length);

  if (DBG_LV2){
   printf('left: %d, ', left);
   printf('right: %d, ', right);
   printf('parent: %d\n', parent);
   printArray(heap, length);
  }

  insertMinHeap(heap, parent, &length);

  printf('length: %d\n\n', length);
  //--*length;

 }

 if (DBG_LV1) printArray(heap, num_letters);

 return heapExtractMin(heap, &length);
}

void buildMinHeap(int array[], int heapSize){

 int index;

 for (index = DIV_BY_2(heapSize); index >= ROOT_INDEX; --index){

  minHeapify(array, index, heapSize);

 }

}

void minHeapify(int array[], int index, int heapSize){

 int left, right, smallest;

 smallest = index;
 left = LEFT(index);
 right = RIGHT(index);

 // Find smallest value
 if (left < heapSize && array[left] < array[index])
  smallest = left;

 if (right < heapSize && array[right] < array[smallest])
  smallest = right;

 if (smallest != index){

  // Exchange
  exchange(&array[index], &array[smallest]);

  // Rebuild heap region
  minHeapify(array, smallest, heapSize);
 }

}

int heapMin(int array[]){
 return array[ROOT_INDEX];
}

int heapExtractMin(int array[], int *length){
 if (DBG_LV0) printf('heapExtractMin()\n');

 if (length < 0){
  printf('[X] Error: heap overflow!\n');
  return -1;
 }

 --*length;

 int heapSize = *length;

 int min = array[ROOT_INDEX];

 --heapSize;

 printf('exchange: array[%d]: %d, array[%d]:%d \n',
     ROOT_INDEX, array[ROOT_INDEX],
     heapSize, array[heapSize]);

 exchange(&array[ROOT_INDEX], &array[heapSize]);

 --heapSize;

 minHeapify(array, ROOT_INDEX, heapSize);

 return min;
}

void insertMinHeap(int array[], int value, int *length){

 if (DBG_LV0) printf('insertMinHeap(value: %d, length: %d)\n',
           value, *length);

 int heapSize = *length;

 ++*length;

 array[heapSize] = INT_MAX;

 if (value > array[heapSize]){
  printf('[X] Error: new value is bigger than biggest element!\n');
 }else{

  array[heapSize] = value;
  if (DBG_LV2) printArray(array, *length);

  while (heapSize > ROOT_INDEX &&
      array[PARENT(heapSize)] > array[heapSize]){

   exchange(&array[heapSize], &array[PARENT(heapSize)]);

   heapSize = PARENT(heapSize);

  }

 }

}

void exchange(int *a, int *b){
  if (DBG_LV3) printf('exchange()\n');

  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

void printArray(int array[], int length){
 if (DBG_LV0) printf('printArray()\n');

 int i;

 for (i = 0; i < length; ++i)
  printf('[%d]', i);

 printf('\n');

 for (i = 0; i < length; ++i)
  printf(' %d ', array[i]);

 if (DBG_LV1) printf('\n\n');
}

int isLeft(struct Data array[], int indexLeft, int indexRight){
 if (DBG_LV0) printf('isLeft()\n');

 if (array[indexLeft].frequency == array[indexRight].frequency){

    if (array[indexLeft].letter < array[indexRight].letter){
     return DIR_LEFT;
    }else{
     return DIR_RIGHT;
    }

 }else if (array[indexLeft].frequency < array[indexRight].frequency){
  return DIR_LEFT;
 }else{
  return DIR_RIGHT;
 }

}

void test1(){
 if (DBG_LV1) printf('test1()\n');

 int i, length = 5;

 if(DBG_LV1) printf('BUILD HEAP ARRAY:\n');
 for(i = 0; i < length; ++i){
  heap[i] = i * 2;
  printf('heap[%d]: %d \n', i, heap[i]);
 }
 heap[4] = 1;

 buildMinHeap(heap, length);

 printArray(heap, length);

 insertMinHeap(heap, 3, &length);

 printArray(heap, length);
}

void test2(){

 int heapLen = num_letters;

 printf('heapLen: %d\n', heapLen);

 printArray(heap, heapLen);

 int result1, result2;

 printf('TEST HEAPEXTRACTMIN:\n');
 result1 = heapExtractMin(heap, &heapLen);
  printf('result1: %d\n', result1);
 printArray(heap, heapLen);

 printf('heapLen: %d\n', heapLen);

 result2 = heapExtractMin(heap, &heapLen);
  printf('result2: %d\n', result2);
 printArray(heap, heapLen);

 printf('heapLen: %d\n', heapLen);

}
```

input.txt:

```
1
2
3
4
1 0 0
1 0 1
1 1
0
```

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