Oberon Ilano

CS 430

Test 3 Branch and Bound

1.

Procedure BranchBound()

integer i, next

float wt, pr

struct job Left\_Child, Right\_Child, Current

Current.UBound CalUB(0, 0, 0)

Current.LBound CalLB(0, 0, 0)

Current.flag -1

Current.j 0

i 1

while Current.j <> totalJobs do

next Current.j + 1

Right\_Child.UBound CalUB(wt, pr, next)

Right\_Child.LBound CalLB(wt, pr, next)

Right\_Child.flag 0

Right\_Child.j next

Left\_Child.flag 1

Left\_Child.j next

If wt + deadline[next] <= totalDeadline then

Left\_Child.UBound CalUB(wt + deadline(next), pr – P(next), next)

Left\_Child.LBound CalLB(wt + deadline(next), pr – P(next), next)

else

Current.UBound pr

Left\_Child.LBound pr

If Left\_Child.LBound <= Right\_Child.LBound AND Left\_Child.UBound <= Right\_Child.UBound then

Current Left\_Child

else

Current Right\_Child

Completed(i) Current.flag

i i + i

if Current.flag = 1 then

pr pr - P(Current.j)

wt wt + deadline(Current.j)

for i 1 to totalJobs increment by 1 do

printf(completed(i))

if completed(i) = 0) then

printf(-(Current.UBound))

end BranchBound

2.

Data Files

Deadline & Penalty

Decreasing Order:

10 12

9 11

8 10

7 9

6 8

5 7

4 6

3 5

2 4

1 3

Duplicate:

1 2

31 3

31 3

19 4

19 4

22 5

38 6

9 7

28 8

25 9

Even:

2 10

8 8

10 6

20 4

4 2

6 12

12 14

18 16

14 18

16 20

Nondecreasing:

1 3

2 4

3 5

4 6

5 7

6 8

7 9

8 10

9 11

10 12

Random:

1 1

3 3

6 6

8 8

10 10

5 5



3.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author(s): Oberon Ilano, F

Assignment: Test 3, Branch and Bound

Due Date: October 25, 2019

Course: CS430

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

//GLOBAL VARIABLES //

int totalDeadline, totalJobs = 1;

int P[100], deadline[100], completed[100]= {0};

struct job{

int j, flag;

float UBound, LBound;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: menu

Author(s): Oberon Ilano

Purpose: Use to select option from the keyboard

Incoming: None

Outgoing: main menu option (char)

Return: None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void menu(){

//option selection

printf("Branch And Bound Method \n");

printf("1. Nondecreasing order items\n");

printf("2. Decreasing order items\n");

printf("3. Duplicate items\n");

printf("4. Even random numbers\n");

printf("5. Random numbers\n");

printf("\nEnter a valid choice: ");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: OpenFile

Author(s): Oberon Ilano

Incoming: numbers (int), char filename

Outgoing: array of random numbers(int)

Return: None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void OpenFile(char \*filename){

FILE \*inFile = fopen(filename, "r+");

//FILE \*inFile2 = fopen("Random.txt", "r+");

if(inFile == NULL){

printf("File Error\n");

exit(1);

}

// read file for random numbers

while( fscanf(inFile, "%d", &deadline[totalJobs]) == 1 || totalJobs == 100){

fscanf(inFile, "%d", &P[totalJobs]);

totalJobs = totalJobs + 1;

totalDeadline += deadline[totalJobs];

}

fclose(inFile);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: CalUB

Author(s): https://www.academia.edu/34852276/

Title\_Implementation\_of\_0-1\_knapsack\_

problem\_using\_branch\_and\_bound\_approach

Return: float cp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float CalUB(float CurrentDL, float CurrentP, int CurrentJob){

float cw = CurrentDL;

float cp = CurrentP;

int i;

for(i = CurrentJob + 1; i <= totalJobs; i++){

if(cw + deadline[i] <= totalDeadline){

cw = cw + deadline[i];

cp = cp - P[i];

}

}

return cp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: CalB

Author(s): https://www.academia.edu/34852276/

Title\_Implementation\_of\_0-1\_knapsack\_

problem\_using\_branch\_and\_bound\_approach

Return: float cp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float CalLB(float CurrentDL, float CurrentP, int CurrentJob){

float cw = CurrentDL;

float cp = CurrentP;

int i;

for(i = CurrentJob + 1; i <= totalJobs; i++){

cw = cw + deadline[i];

if(cw < totalDeadline)

cp = cp - P[i];

else

return (cp - (1 - (cw - totalDeadline) / deadline[i]) \* P[i]);

}

return cp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: sort

Author(s): https://www.academia.edu/34852276/

Title\_Implementation\_of\_0-1\_knapsack\_

problem\_using\_branch\_and\_bound\_approach

Incoming: numbers (int), char filename

Outgoing: array of random numbers(int)

Return: None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Sort(){

int i, j, temp;

for(i = 1; i <= totalJobs; i++){

for(j = i + 1; j <= totalJobs; j++){

if(deadline[j] < deadline[j - 1]){

temp = P[j];

P[j] = P[j-1];

P[j-1] = temp;

temp = deadline[j];

deadline[j] = deadline[j - 1];

deadline[j - 1] = temp;

}

}

}

for(i = 1; i <= totalJobs; i++){

for(j = i + 1; j <= totalJobs; j++){

if(P[j] > P[j - 1]){

temp = P[j];

P[j] = P[j - 1];

P[j - 1] = temp;

temp = P[j];

P[j] = P[j - 1];

P[j - 1] = temp;

temp = deadline[j];

deadline[j] = deadline[j - 1];

deadline[j - 1] = temp;

}

}

}

printf("Jobs: \t\t");

for(i = 1; i <= totalJobs; i++)

printf("%d\t", i);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Definition#

Name: BranchBound

Author(s): https://www.academia.edu/34852276/

Title\_Implementation\_of\_0-1\_knapsack\_

problem\_using\_branch\_and\_bound\_approach

Incoming: numbers (int), char filename

Outgoing: array of random numbers(int)

Return: None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void BranchBound(){

int i, next;

float wt = 0, pr = 0;

struct job Left\_Child, Right\_Child, Current;

Current.UBound = CalUB(0, 0, 0);

Current.LBound = CalLB(0, 0, 0);

Current.flag = -1;

Current.j = 0;

i = 1;

while(Current.j != totalJobs){

next = Current.j + 1;

Right\_Child.UBound = CalUB(wt, pr, next);

Right\_Child.LBound = CalLB(wt, pr, next);

Right\_Child.flag = 0;

Right\_Child.j = next;

Left\_Child.flag = 1;

Left\_Child.j = next;

if(wt + deadline[next] <= totalDeadline){

Left\_Child.UBound = CalUB(wt + deadline[next],pr - P[next],next);

Left\_Child.LBound = CalLB(wt + deadline[next],pr - P[next],next);

}

else{

Current.UBound = pr;

Left\_Child.LBound = pr;

}

if(Left\_Child.LBound <= Right\_Child.LBound && Left\_Child.UBound <= Right\_Child.UBound)

Current = Left\_Child;

else

Current = Right\_Child;

completed[i] = Current.flag;

i++;

if(Current.flag == 1 ){

pr = pr - P[(Current.j)];

wt = wt + deadline[(Current.j)];

}

}

printf("\nCompleted: \t");

for(i = 1;i<= totalJobs;i++)

printf("%d\t", completed[i]);

if(completed[i] == 0)

printf("\nPenalty : %f\n", -(Current.UBound));

}

// MAIN DRIVER //

int main(){

char choice = ' ';

menu();

scanf("%c", &choice);

switch(choice){

case '1': printf("\nNon-decreasing order items\n");

OpenFile("Nondecreasing.txt"); break;

case '2': printf("\nDecreasing order items\n");

OpenFile("Decreasing.txt");break;

case '3': printf("\nDuplicate Items\n");

OpenFile("Duplicate.txt"); break;

case '4': printf("\nEven random items\n");

OpenFile("Even.txt");break;

case '5':printf("\nRandom Numbers\n");

OpenFile("Random.txt");break;

default:

printf("\nInvalid Choice!\n");break;

}

Sort();

BranchBound();

return 0;

}



4.

The time complexity for sorting the data is .

Solving the upper bound takes and lower bound takes

Therefore, the worst-case complexity takes +.