Exploration and Practice of Teaching Method on Computer Algorithm

Xiang Wang
School of Information and Electronic Engineering
Tianjin Vocational Institute
Tianjin, China
E-mail: wangxiang7504@163.com

***Abstract*—Algorithm course is a key component in computer science-related majors of higher schools. Many college computer science teachers lay emphasis on the improving the teaching methods of algorithms. Subject construction is the foundation of collegiate specialty construction. This paper talks about the reformation and innovation of the teaching methods of computer algorithm to improve the teaching efficiency of algorithms course through teaching the process of greedy algorithms.**

Keywords—algorithms course; greedy algorithm; teaching method

I. INTRODUCTION

An algorithm is a cornerstone of computer science. In fact, computer algorithm occupies a crucial position within the development of the information technology, especially in the mobile Internet era of regulated cloud technology and big data. However, in recent years, computer science and relevance special field students pay less attention to algorithm courses. They think that, for students majoring in computer science, there is no need to learn the principles of algorithm design as long as they do well in the area of programming. The reason leading to such a situation lies in many different aspects, among which is the problems in teaching methods of algorithm courses. Greedy algorithm is the important content of computer algorithm system. I would like to discuss with you issues concerning the reformation and innovation of teaching methods of computer algorithm.

II. THE INTRODUCTION OF GREEDY ALGORITHM

Straightforward concept introduction makes students lack perceptual knowledge due to how abstract greedy algorithms can be. Therefore, the students can't understand the connotations and method of thinking of greedy algorithms. Teachers of algorithms may start their teaching activities with simple cases to make the students understand the connotation of algorithm. For example, the problem of "Making Change" is a good example of a greedy algorithm.

Let's suppose the currency denominations include 50, 20, 10, 5 and 1 Yuan, which devises a method to pay amount R to customer using fewest number of coins. The problem is to make change of a given amount using the smallest possible number of coins. That is, we need to make change for *n* units using the least possible number of coins. Greedy algorithms work by making the decision that seems most promising at any moment; it never reconsiders this decision no matter what situation may arise later. For example, if we want make change for 98 RMB, here R =98 and the solution contains one 50-yuan bill, two 20-yuan bills, one 5-yuan bill and 3 1-yuan bills . The reason such an algorithm is called greedy is because, at every stage, it chooses the largest coin without worrying about the consequences. Moreover, it never changes strategy, and once a coin has been included in the solution set, it remains there. To solve the problem of "Making Change" in an optimal way, the algorithm needs to maintain two sets. The first set contains chosen items and the second set contains rejected items. Let's look at the function that is used to solve the problem of "Making Change". The function of "Making Change" is as follows:

```
int Greedy1(int R)
{
  int i, j=0, n[7], m[7]={100,50,20,10,5,2,1};
  for(i=0;i<7;i++)
            n[i]=0;
  i=0;
  while(R>0){
            if(m[i]<=R){
                      R-=m[i];
                      n[i]++;
            }
            else i++;
  }
  return *n;
}
```

We can also set the amount of various denominations of the money in algorithmic routine to accord with practical applications. It can be expressed as k[1…n]. The improved program is as follows:

```
int Greedy2(int R)
{
  int i, j=0,n[7], m[7]={100,50,20,10,5,2,1};
  int k[7]={500,400,300,200,100,50,40};
  for(i=0;i<7;i++)
              n[i]=0;
  i=0;
  while(R>0){
              if((m[i]<=R)&&(k[i]>=1)){
                          R-=m[i];
                          n[i]++;
                          k[i]--;
              }
               else i++;
  }
  return *n;
}
```

In the teaching process, we should let students understand the different execution steps by analyzing the algorithmic routines before we introduce the concrete notion of greedy algorithm. In fact, greedy algorithm consists of four key points:

- The first key point is to check whether the chosen set of items provides a solution.
- The second key point is to check the feasibility of a set.
- The third key point is to find which of the candidates is the most promising.
- The fourth key point is to give the value of a solution.

Then, we may let students learn the strict definition of a greedy algorithm on the basis of a large number of perceptual knowledge. This cultivates their rational thinking and helps develop their abilities to solve problems by using algorithmic routine (i.e. a greedy algorithm is a mathematical process to make the locally optimal choice at each stage with the hope of finding a global optimum).

III. The Mutual Integration of Algorithms Course and the Other Rrelated Courses in Computer Science

If we do not pay attention to the mutual integration of algorithms course and the other related courses, we neglect the basic and instrumental role of algorithms course. Algorithm teaching is merely a castle in the air without the deep integration of algorithms course and computer specialty course system. If our students do not feel strongly that algorithm courses are important to the success of their specialized study, they may lose that "spark", their inspiration, and their passion. In fact, we shall focus on guiding students to apply the basic idea and principle of computer algorithms to the software engineering practice. For example, greedy algorithm play an important role in the problem of scheduling on multi-machines. Let T (n) be time complexity of matrix multiplication, we can explain the T(n) through the following recursive expression i.e.,

```
int Dri2015()
{  int n=7,m=3,t[]={6,17,5,19,8,5,4};
   //jobs to be allocated
   Greedy3(t,n,m);
  return 0; }
void Greedy3(int t[],int n,int m)  {
     int fn,fm, M[]={0,0,0,0,0,0,0};
     for(int i=0;i<n;i++) {
           int max=0,min=10000;
           fn=0; fm=0;
           for(int j=0;j<n;j++) {
              if(max<t[j]){
                   max=t[j]; fn=j;   } }
           for(j=0;j<m;j++)  {
              if(M[fm]>M[j])   fm=j; }
     M[fm]=M[fm]+t[fn];  t[fn]=0;
     cout<<fn<<"work "<<fm<<"machine"<<endl; } }
```

Of course, we can also think about the more general case of the problem of scheduling on multi-machines. To make it easier for us to carry out teaching activities, we can spell out to the students how to solve the problem of scheduling on multi-machines by greedy algorithms through the following example. Let *m* be the number of the machines, let *n* be the number of the jobs, let sort2015 be the name of the sorting function.

```
#define N 10
typedef struct node{
            int ID,time;
}proc2015;
typedef struct Node{
            int ID,a2015;
}m2015;
m2015 con2015[N];proc2015 job[N];
m2015* Find_min(m2015 a[],int m){
m2015* tp2015=&a[0];
for(int i=1;i<m;i++){
    if(a[i].a2015<tp2015->a2015)
    tp2015=&a[i];
 }
 return tp2015;
}
void scheduler2015(int n, int m, int tp2015){
m2015* num2015;
 ......
 sort2015(job,n);
 for(i=0;i<n;i++){
      num2015=Find_min(con2015,m);
      printf("...%d...%d...%d...: %d\n",num2015->ID,num2015-
>a2015,num2015->a2015+job[i].time,job[i].ID);
       num2015->a2015+=job[i].time;
  }
  tp2015=con2015[0].a2015;
  for(i=1;i<m;i++){
       if(con2015[i].a2015>tp2015)
       tp2015=con2015[i].a2015;
  }
  ......
 printf("...: %d\n",tp2015);
}
```

IV. Teaching Direction of the Theoretical Analysis of Greedy Algorithm

The study of computer algorithm theory is very important to improve students' professional quality on computer science, and is an important guarantee of the sustainable

development of the students. In the teaching process, we cannot solely focus on the algorithm technique. Algorithm teaching should focus not only on the algorithm technique but also on developing students' comprehensive analysis capability. However, it's worth noting that we should choose moderately difficult material concerning algorithms applications in teaching English. In order to find out the way to utilize the teaching materials for the algorithm theory effectively, teachers should operate the guidelines step-by-step to make the students' tasks easier. For example, analysis of the algorithm function of the knapsack problem is the proper teaching material to analyze the features and advantages of the algorithm theory.

Let set M be the total weight, let *n* be the number of the articles, let W[i] be the weight of the (i)th article, set P[i] to the value of the (i)th article, set X[i] to the (i)th component of the vector of n elements, and let sort2015 be the name of the sorting function. Therefore, the following program can be taken as an example.

```
void knapsack(int N2015, float M2015, float W[], float P[], float
X[]){
 sort2015(n,W,P);
 int i;
 float m=M2015;
                              for(i=0;i<=n-1;i++)
        X[i]=0;
         i=1;
         while(W[i]<=m){
            X[i]=1;
            m-=W[i];
            i++;
         }
         if(i<=n-1) X[i]=m/W[i];
 }
```

As we all know, it is very difficult to make students understand the correctness of the function above. We 'd better adopt apagoge to solve the problem. Let solution of the knapsack problem be $(X_1,X_2,\ldots,X_n)$, and $\sum_{i=1}^{n} W_i X_i = M$, where $0 \leqslant X_i \leqslant 1$. If W1≤M, then X1=1, otherwise X1=M/ W1.

First of all, If n=1, proposition clearly establish. Secondly, If n<m, proposition clearly establish. When n=m, if the proposition is not established, there certainly exists another optimal solution $(Y_1,Y_2,\ldots,Y_n)$, and $Y_i<X_i$, $\sum_{i=1}^{n} W_i Y_i = M$, $\sum_{i=1}^{n} Y_i P_i > \sum_{i=1}^{n} X_i P_i$. $Y_1,Y_2,\ldots,Y_n$ should not be all zeros, and $\sum_{i=2}^{n} Y_i W_i > (X_i - Y_i) W_1$. We can guide the students to build another solution $(Z_1,Z_2,\ldots,Z_n)$, it follows that

$Z_1=X_1$, $\sum_{i=2}^{n} (Y_i - Z_i) W_i = (Z_i - Y_i) W_1 = (X_i - Y_i) W_1$.

Because $(P_1/W_1, P_2/W_2, \ldots, P_n/W_n)$ is in descending order, thus $\sum_{i=1}^{n} Z_i P_i > \sum_{i=1}^{n} Y_i P_i$, i.e., $(Z_1,Z_2,\ldots,Z_n)$ is also the optimal solution. Because $Z_1=X_1$, the problem may boil down to: $(W_2,\ldots,W_n)$, $(P_2,\ldots,P_n)$. According to the induction hypothesis, $Z_2=X_2$, then we have $Z_3=X_3$,, …, $Z_n=X_n$, for the same reason, i.e.,

$\sum_{i=1}^{n} Z_i P_i = \sum_{i=1}^{n} Y_i P_i$ is contradictory to $\sum_{i=1}^{n} Z_i P_i > \sum_{i=1}^{n} Y_i P_i > \sum_{i=1}^{n} X_i P_i$.

REFERENCES

[1] Liu Jing. An Introduction to Computer Algorithm—Techniques of Design and Analysis[M]. Beijing: Science Press, 2003.

[2] Deng Xiangyang, WanTingting. Design and Analysis of Algorithms[M]. Beijing: Metallurgical Industry Press, 2006.

[3] Xiang, Wang. " Improvement of Teaching Method of Greedy Algorithm for Knapsack Problem ". In Proceedings of the 2nd International Conference on Computer Science and Service System(CSSS 2012). ISBN : 978-1-4673-0719-2. 2012 ,8.

[4] Aho, A.V., Hopcroft, J.E.,and Ullman, J.D.(2007). The Design and Analysis of Computer Algorithms(Huang, L.P., Wang, D.J., and Zhang, S., Trans.). Beijing, China: China Machine Press.(1974).

[5] Wang, X.D.(2001). The Design and Analysis of Computer Algorithms. Beijing, China: Publishing House of Electronics Industry.

[6] Liu, J.(2003). An Introduction to Computer Algorithm—Techniques of Design and Analysis. Beijing, China:Science Press.

[7] Su, D.F.,and Zhong, C.(2001).The Design and Analysis of Computer Algorithms. Beijing, China: Publishing House of Electronics Industry.

[8] E.Horowitz.,and S.Sahni.(1978). Fundamentals of Computer Algorithms. U.S.A.: Compter Science Press.

[9] E.Horowitz., S.Sahni.,and S.Rajasekaran.(1996). Computer Algorithms/C++. U.S.A.: Compter Science Press.

[10] T.H.Cormen., C.Leiserson.,and C.Stein.(2001). Introduction to Algorithms. U.S.A:The MIT Press.

[11] Alfred V.Aho.,John E. Hopcroft., and Jeffrey D.Ullman.(1983). Data Structures and and Algorithms. U.S.A: Addison-Wesley.

[12] Sara Baase., and A.Gelder.(2000). Computer Algorithms:Introduction to Design and Analysis. U.S.A: Addison-Wesley.