LECTURE 10:

The Greedy approach

Optimization problems

The general structure of an optimization problem is:

Find x in X such that:

(i) x satisfies some constraints

(ii) x optimizes (minimizes or maximizes) a criterion

Particular case:

X is a finite set – the problem is a combinatorial optimization problem

Not all such problems are easy to solve.

Optimization problems

Example 1.

Let $A=\{a_1,\ldots,a_n\}$ and $m<n$
Find a subset S of A such that:

(i) The cardinal of S is m (constraint)
(ii) The sum of elements in S is maximal (optimization criterion)

Remark. X = the set of all 2^n subsets of A
A brute force approach is of $O(2^n)$.

Optimization problems

Example 2 (subset sum).

Let $A=\{a_1,\ldots,a_n\}$ and $m<n$
Find a subset S of A such that:

(i) The sum of elements in S is C ($C< a_1+\ldots+a_n$) (constraint)
(constrained satisfaction problem)
(ii) The number of elements is minimal (optimization criterion)
(optimization problem)

Remark. X = the set of all 2^n subsets of A
A brute force approach is of $O(2^n)$. A more efficient method should be found …

Outline

- Optimization problems
- Basic idea of greedy technique
- Examples
- Correctness verification and efficiency analysis
- Some classical applications

Basic idea of greedy technique

Let's reformulate the optimization problem as follows:

Let $A=(a_1,\ldots,a_n)$ be a multiset (a set of not necessarily distinct elements). Find $S=(s_1,\ldots,s_k)$such that S satisfies some constraints and optimizes a certain criterion.

Basic idea of greedy technique:

- S is constructed successively starting with the first element
- At each step a new element (that element which seems to be the best at that moment) is selected from A.
- Once a choice is made it is final (the greedy approach at each step takes the currently best element, without regard for future consequences; there are no back steps to make corrections)

Basic idea of greedy technique

The general structure of a greedy algorithm:

```
Greedy(A)
  S←Ø
  WHILE "S is not completed" AND "there exist unselected elements in A"
        DO
     "choose the best currently available element a from A"
     IF "by adding a to S the constraints are satisfied"
     THEN "add a to S"
  RETURN S
```

- Remark: You take the best you can get right now, without regard for future consequences
- You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

Basic idea of greedy technique

The most important part of a greedy algorithm is the selection of an element at each step.

The elements are selected based on a selection criterion which is established depending on the problem.

The selection criterion is frequently based on some heuristics (= technique based on experiential data and intuition rather than on theoretical analysis)

Outline

- Optimization problems
- Basic idea of greedy technique
- Examples
- Correctness verification and efficiency analysis
- Some classical applications

Examples

Problem 1 (maximal subset sum of a given cardinal)

Find a subset S of a finite multiset A such that:

(i) S has m<=card A elements

(ii) The sum of elements in S is maximal

Example:

Let A={5,1,7,5,4} and m=3.

Then S={5,5,7}

Examples

Greedy approach: (partially) sort decreasingly the elements of A and select the first m elements

```
Subset(A[1..n],m)  //variant1
FOR i ← 1,m DO
  k ← i
  FOR j ← i+1,n DO
     IF A[k]<A[j] THEN k ← j  ENDIF
  ENDFOR
  IF k<>i THEN A[k]↔A[i] ENDIF
  S[i] ← A[i]
ENDFOR
RETURN S[1..m]
```

```
Subset(A[1..n],m)  //variant2
A[1..n] ← decreasing_sort(A[1..n])
FOR i ← 1,m DO
     S[i] ← A[i]
RETURN S[1..m]

// less efficient than variant 1 (if A is
        not already decreasingly
        sorted)
```

Remark. One can prove that for this problem the greedy strategy always produces an optimal solution

Examples

Problem 2 (coin changing problem)

Let us suppose that we have an unlimited number of coins of the following values: $\{v_1, v_2, \ldots, v_n\}$. Find a way to cover an amount C such that the number of used coins is minimal

Let's denote by s_i the number of coins of value v_i

Constraint: $s_1v_1+\ldots+s_nv_n=C$

Optimization criterion: $s_1+s_2+\ldots+s_n$ is minimal

Greedy approach: starting from the coin of the largest value try to cover as much as possible from the initial amount C; continue by choosing the next largest value and so on …

Examples

```
Coins(v[1..n],C)
v[1..n] ← decreasing_sort(v[1..n])
FOR i ← 1,n DO S[i] ← 0 ENDFOR
i ← 1
WHILE C>0 and i<=n DO
   S[i] ← C DIV v[i]      // maximal number of coins of value v[i]
   C ← C MOD v[i]         // remaining amount
   i ← i+1
ENDWHILE
IF C=0 THEN RETURN S[1..n]
          ELSE “the problem has no solution”
ENDIF
```

Examples

Remarks:

1. Sometimes the problem has no solution:

Example: V=(20,10,5) and C=17

However if we have coins of value 1 then the problem always has a solution

1. 2. Sometimes the greedy approach doesn't give an optimal solution

Example: V=(25,20,10,5,1), C=40

Greedy approach: (1,0,1,1,0)

Non-greedy approach: (0,2,0,0,0)

A sufficient condition for optimality is: $v_1>v_2>\ldots>v_n=1$ and $v_{i-1}=d_{i-1}v_i$

Examples

So the greedy approach leads to:

- simple and intuitive algorithms
- efficient algorithms

But

- it does not always lead to an optimal solution (a local greedy choice can have negative global consequences)

- sometimes the non-optimal solutions are close to optimal ones (an algorithm which gives near-optimal solutions is called approximation algorithm)

Since the greedy approach doesn't always ensure the optimality of the solution we have to analyze for each particular problem if the algorithm leads to an optimal solution

Outline

- Optimization problems
- Basic idea of greedy technique
- Examples
- Correctness verification and complexity analysis
- Some classical applications

Correctness verification

There is no general method to prove that a greedy method yields an optimal solution (in fact in most of practical situations it gives only sub-optimal solutions).

However many of the problems for which the greedy solution is optimal share the following properties:

- the optimal substructure property
- the greedy choice property

Optimal substructure property

When can we say that a problem has the optimal substructure property ?

When … for an optimal solution $S=(s_1,\ldots,s_k)$ of a problem of size n the subset $S_{(2)}=(s_2,\ldots,s_k)$ is an optimal solution of a subproblem of size (n-1).

How can we verify if a problem has this property ?

Using a proof by contradiction

Greedy choice property

When can we say that a problem has the greedy choice property ?

When … an optimal solution either is constructed by a greedy strategy or can be transformed into an optimal solution whose elements are chosen by a greedy strategy

How can we verify if a problem has this property ?

First we prove that by replacing the first element of an optimal solution with an element selected by the greedy strategy the solution remains optimal. Then we prove the same for the other elements by using the mathematical induction or by using the optimal substructure property.

Correctness verification

Example: (maximal sum subset of fixed cardinal)

Let $A=(a_1>=a_2>=\ldots>=a_n)$. The greedy solution is $(a_1,\ldots,a_m)$.

Let $O=(o_1,\ldots,o_m)$ be an optimal solution.

a) Greedy choice property. Suppose that $o_1<>a_1$. This means that $o_1<a_1$ Then $O'=(a_1,o_2,\ldots,o_m)$ has the property:

$$a_1+o_2+\ldots+o_m> o_1+o_2+\ldots+o_m$$

This means that O' is better than O. This contradicts the fact that O is optimal. Thus o_1 has to be a_1

b) Optimal substructure property. Proof by contradiction: suppose that $(o_2,\ldots,o_m)$ is not optimal for $A_{(2)}=(a_2,\ldots,a_n)$. Let us consider that $O'_{(2)}= (o'_2,\ldots,o'_m)$ is an optimal solution of the subproblem. Then $O'=(a_1, o'_2,\ldots,o'_m)$ is a better solution than O. Contradiction…thus the problem has the optimal substructure property.

Correctness verification

Example: coin changing problem

Let $V=(v_1>v_2>\ldots>v_n=1)$ the values of the coins and $v_{i-1}=d_{i-1}\ v_i$. The greedy solution, $(g_1,\ldots,g_m)$, is characterized by $g_1 = C \text{ DIV } v_1$

Let $O=(o_1,\ldots,o_m)$ be an optimal solution.

a) Greedy choice property. Let us suppose that $o_1<g_1$. Then the amount $C'=(C \text{ DIV } v_1-o_1)v_1$ is covered by smaller coins. Due to the property of coin values, by replacing o_1 with g_1 one obtains a better solution (the same value can be covered by fewer coins of higher value). Thus the problem has the greedy choice property.

b) Optimal substructure property. Easy to prove.

Complexity analysis

Greedy algorithms are efficient

Usually the dominant operation is the selection of a new element or the sorting of the set A

Thus the complexity of greedy analysis is

$O(n^2)$ or $O(n \lg n)$ or even $O(n)$ (for particular sets of values the sorting process can be of linear complexity)

Outline

- Optimization problems
- Basic idea of greedy technique
- Examples
- Correctness verification and complexity analysis
- Some classical applications

Some classical applications

The knapsack problem

Let us consider a set of objects. Each object is characterized by its weight w and a value v. We want to fill in a knapsack of capacity C such that the total value of selected objects is maximal.

Variants:

(i) Continuous variant: entire objects or fractions of objects can be selected. The components of the solution are from [0,1].

(ii) Discrete variant (0-1): an object is either entirely transferred into the knapsack or it is not transferred at all.

Some classical applications

The knapsack problem - motivation

The knapsack-problem (discrete variant) was used as the basis for a cryptographic system (that has since been broken).

Any constraint satisfaction problem where the solution depends on selecting some items whose values are maximized (or minimized), and whose weight is equal to a target value is related to the knapsack problem.

Knapsack related problems are encountered in numerous industrial domains such as transportation, logistics, cutting and packing, telecommunication, reliability, advertisement, investment, budget allocation, and production management

The knapsack problem

Example:

Value	Weight	Relative profit (value per weight)
6	2	3
5	1	5
12	3	4

C=5

Selection criteria:

Increasing order of the weight (put as many objects as possible):

5+6+12*2/3=11+8=19

The knapsack problem

Example:

Value	Weight	Relative profit (value per weight)
6	2	3
5	1	5
12	3	4

C=5

Selection criteria:

Increasing order of the weight (put as many objects as possible):
5+6+12*2/3=11+8=19

Decreasing order of the value (put the most valuable objects):
12+6=18

The knapsack problem

Example:

Value	Weight	Relative profit (value per weight)
6	2	3
5	1	5
12	3	4

C=5

Selection criteria:

Increasing order of the weight (put as many objects as possible):
5+6+12*2/3=11+8=19

Decreasing order of the value (put the most valuable objects):
12+6=18

Decreasing order of the relative profit (put objects which are small and valuable first):
5+12+6*1/2=17+3=20

The knapsack problem

```
Knapsack(w[1..n],v[1..n])
 "sort w and v decreasingly by the relative profit"
  FOR i ← 1,n DO S[i] ← 0 ENDFOR
  i ← 1
  WHILE C>0 AND i<=n DO
       IF C>=w[i] THEN   S[i] ← 1
                         C ← C-w[i]
                  ELSE   S[i] ← C/w[i]
                         C ← 0
       ENDIF
       i ← i+1
   ENDWHILE
  RETURN S[1..n]
```

The knapsack problem

Correctness verification:

for the continuous variant of the knapsack problem, the greedy approach leads to an optimal solution

Remarks:

- A greedy solution satisfies: S=(1,1,…,1,f,0,..,0)

$s_1 w_1+\ldots+s_n w_n=C$ (the equality restriction can be always satisfied)

- The objects are decreasingly sorted by the relative profit:

$v_1/w_1>v_2/w_2>\ldots>w_n/w_n$

Proof.

Let $O=(o_1,o_2,\ldots,o_n)$ an optimal solution. We shall prove by contradiction that it is a greedy solution. Let us suppose that O is not a greedy solution and let us consider a greedy solution $O'=(o'_1,o'_2,\ldots,o'_n)$

The knapsack problem

Let $B_+ = \{i | o'_i >= o_i\}$ and $B_- = \{i | o'_i < o_i\}$, k – the smallest index for which $o'_i < o_i$.

Due to the structure of a greedy solution it follows that any index i from B+ is less than any j from B_.

On the other hand both solutions satisfy the restriction, thus:

$$o_1 w_1 + \ldots + o_n w_n = o'_1 w_1 + \ldots + o'_n w_n$$

$$\sum_{i \in B+} (o'_i - o_i) w_i = \sum_{i \in B_-} (o_i - o'_i) w_i$$

$$P' - P = \sum_{i=1}^{n} (o'_i - o_i) v_i = \sum_{i \in B+} (o'_i - o_i) w_i \frac{v_i}{w_i} - \sum_{i \in B_-} (o_i - o'_i) w_i \frac{v_i}{w_i}$$

$$P' - P \geq \frac{v_k}{w_k} \sum_{i \in B+} (o'_i - o_i) w_i - \frac{v_k}{w_k} \sum_{i \in B+} (o_i - o'_i) v_i = 0$$

Thus the greedy solution is at least as good as O (which is supposed to be optimal). The optimal substructure property is easy to prove.

Activity selection problem

Let $A=\{a_1,\ldots,a_n\}$ be a set of activities which share the same resource. Each activity a_i needs a time interval $[s_i,f_i)$ to be executed. Two activities are considered compatible if their associated time intervals are disjoint.

The problem asks us to select a maximal number of compatible activities.

Remarks. There are different criteria of selecting the activities

- In increasing order of the ending time (optimal)
- In increasing order of the interval length
- In increasing order of the starting time

Activity selection problem

```
// each element a[i] contains two fields:
//    a[i].s    -  starting time
//    a[i].f    -  ending time

Activity_selection(a[1..n])
a[1..n] ← decreasing_sorting_by_f(a[1..n])
x[1] ← a[1]
k ← 1
FOR i:=2,n DO
  IF a[i].s>=x[i].f THEN
                        k ← k+1
                        x[k] ← a[i]
  ENDIF
ENDFOR
RETURN x[1..k]
```

Activity selection problem

Correctness verification. Let us suppose that the activities are increasingly sorted by the ending time ($a_1.f < a_2.f < \ldots < a_k.f$).

Greedy choice property: Let $(o_1,o_2,\ldots,o_k)=(a_{i1},a_{i2},\ldots,a_{ik})$ be an optimal solution. Since a_1 ends before any other activity it follows that a_{i1} can be replaced with a_1 without altering the compatibility of selected activities or their number.

Optimal substructure property. Let us consider an optimal solution: $(a_1,o_2,\ldots,o_k)$ (based on the previous property it follows that we can consider that $o_1=a_1$). Let us suppose that $(o_2,\ldots,o_k)$ is not an optimal solution for the selection problem corresponding to the subset of activities $\{a_2,a_3,\ldots,a_n\}$. It follows that it would exist another solution $o'=(o'_2,\ldots,o'_{k'})$ characterized by $k'>k$. This would lead to a new solution $(a_1, o'_2,\ldots,o'_{k'})$ which would be better than $(a_1,o_2,\ldots,o_k)$. Contradiction.

Other problems

Other problems for which the greedy technique leads to an optimal solution:

- Design of data-compression codes (Huffman algorithm)
- Find minimum spanning tree (Kruskal and Prim's algorithms)
- Find shortest paths from a single source (Dijkstra algorithm)

There are also a lot of problems for which the greedy approach can be applied, even if it does not guarantee that it produces the optimal:

- Bin packing
- Tasks scheduling

Next lecture will be on …

… dynamic programming strategy

… and its applications