

Greedy algorithms

“Greedy” is a strategy to solve optimization problems. In each step:

- Find a safe choice that leaves optimality possible.
- Form a subproblem (or subproblems), and solve it recursively. This supposedly gives an optimal solution of the subproblem.
- From the solutions to the subproblem, form a solution of the original problem.

The arguments

- The first choice is really safe (Greedy-choice property).
- The solution formed from the solutions of the subproblems is actually optimal (Optimal-substructure).

ALG/TITCS L03-1

General form of proofs

The Greedy Choice Property

- Suppose there is an optimal solution S that does not make the greedy choice.
- Transform S to another solution S' that does make the greedy choice.
- Prove that S' is also optimal.

This implies that whenever there is an optimal solution, we have an optimal solution that makes the greedy choice.

For most optimization problem, the existence of an optimal solution is trivial, so we know there must be an optimal solution that uses the greedy choice.

ALG/TITCS L03-2

General form of proofs (cont'd)

The Optimal Substructure Property

- Suppose the solution S that we formed from the solution S_1 of the subproblem is not optimal.
- Using the greedy choice property, let S' be an optimal solution that does make our greedy choice. Thus S' is a better solution than S .
- Form a solution S'_1 to the subproblem from S .
- Argue that S'_1 is a better solution than S_1 for the subproblem.

Since S_1 is an optimal solution of the subproblem, this leads to a contradiction. The solution must thus be optimal.

ALG/TITCS L03-3

Matroids: introduction

“Greedy” is a strategy. With the same problem, there can be many different greedy algorithms. We do not know whether any of them is optimal, and if so, which ones of them are optimal—until we write a proof.

However, there are some problems that are really similar. They don't just share the fact that they can be solved by a greedy strategy, but actually they share the **same algorithm**, and the **same proof**!

If we can formally say that a new problem in a group of such problem, we don't need to design a greedy algorithm, and we don't need to prove that a newly designed algorithm works. All we need is to reuse the algorithm.

Matroid defines one such *class of problems*. If you know a problem is of a form of matroid, you know how to solve it greedily.

ALG/TITCS L03-4

Matroids: intuitive idea

Matroid gets its name from its strong relation to a **matrix**.

A matrix can be viewed as consisting of some vectors. For example:

$$\begin{pmatrix} 1 & 2 & 2 & 4 \\ 0 & 0 & 1 & 3 \\ 1 & 3 & 4,3 & -5 \\ 2 & 4 & 5 & 11 \end{pmatrix}$$

can be seen as consisting of the vectors $(1, 2, 2, 4)$, $(0, 0, 1, 3)$, $(1, 3, 4, 3, -5)$ and $(2, 4, 5, 11)$.

A matrix is “linearly dependent” if there is one vector that is a linear combination of the other vectors (i.e. sum of non-zero multiples of the other vectors).

For example, the above matrix is linearly dependent, since $(2, 4, 5, 11)$ is the sum of $(0, 0, 1, 3)$ and twice of $(1, 2, 2, 4)$.

A matrix is linearly independent if it is not linearly dependent.

ALG/TITCS L03-5

Matroids: intuitive idea (cont'd)

A subset of the vectors in a matrix forms a submatrix.

Given a matrix consisting of at least one vector, we can always find a submatrix (or submatrices) which is linearly independent.

Let us associate a positive weight to each of the vector. Now the following problem is well defined:

Given a set of rows S and a weight function $w : S \rightarrow \mathbb{R}^+$. Find a set of linearly independent rows $I \subseteq S$ that has the maximum total weight.

This problem can be solved using a greedy algorithm. Surprisingly, the algorithm and the proof can be reused for other problems, provided that they share some simple properties.

ALG/TITCS L03-6

The greedy algorithm

The algorithm to solve the weighted maximum independent submatrix problem is very simple:

```

Sort  $S$  in descending order of weights.
 $I_0 \leftarrow \emptyset$ 
for each  $V \in S$  (from the largest weight to the smallest)
do
  If  $I_0 \cup \{V\}$  is linearly independent
  Then
     $I_0 \leftarrow I_0 \cup \{V\}$ 
Return  $I_0$ 

```

Instead of proving the above algorithm, we will first show the key properties of matrix which make our proof possible, and then show that the algorithm is optimal.

ALG/TITCS L03-7

Matroid: definition

Consider a set S , and a set \mathcal{I} of subsets of S .

We say that the ordered pair (S, \mathcal{I}) is a matroid if it has the following properties. Terminology: we say a subset I of S is independent if $I \in \mathcal{I}$.

- **Finite and non-empty:** S and \mathcal{I} are both finite and non-empty sets.
- **Hereditary:** if a set I is independent, then all subsets of I are independent.
- **Exchange property:** For any two independent sets A and B in S , if $|A| < |B|$, then there is an element x of $B - A$ so that $A \cup \{x\}$ is independent.

E.g.: if S is the vectors in a matrix, and a set of these vector is in \mathcal{I} iff they are linearly independent, then (S, \mathcal{I}) is a matroid.

ALG/TITCS L03-8

The weighted maximum independent subset problem in matroid

Again, we associate a positive weight with each element of S . The weighted maximum independent subset problem in the matroid is defined as follows.

```

Given a matroid  $(S, \mathcal{I})$  and a weight function  $w : S \rightarrow \mathbb{R}^+$ . Find
an independent set  $I \subseteq S$  that has the maximum total weight.

```

Notice that this is a very general problem. An algorithm to this problem means an optimal algorithm for the weighted maximum independent subset problems in any matroid.

E.g. the MST problem, as we will see later, is actually such a problem.

So such an algorithm is much more powerful than any algorithm that we learnt before: it solves a whole class of problems.

The greedy algorithm is simple...

ALG/TITCS L03-9

The greedy algorithm

```

Sort  $S$  in descending order of weights.
 $I_0 \leftarrow \emptyset$ 
for each  $x \in S$  (from the largest weight to the smallest)
do
  If  $I_0 \cup \{x\}$  is independent
  Then
     $I_0 \leftarrow I_0 \cup \{x\}$ 
Return  $I_0$ 

```

The algorithm for the weighted maximum independent submatrix problem can be reused directly here, giving a greedy algorithm.

It is clear that the algorithm always output an independent subset.

We will show that the algorithm is optimal.

ALG/TITCS L03-10

The greedy algorithm is optimal for matroids

To prove the greedy algorithm is optimal, we want to define the greedy choice and the subproblem that we are left with after the greedy choice is made.

The greedy choice: we find the element x of S with the largest weight, and add it to I_0 if this would leave I_0 independent.

The subproblem: we are given a set $S' \subseteq S$ and a subset $I_0 \subseteq S$, where I_0 is independent, and $I_0 \cap S' = \emptyset$. We want to find the subset I' of S' so that $I_0 \cup I'$ is independent, and has the largest total weight among all such subsets.

The original problem in the subproblem form: $S' = S, I_0 = \emptyset$.

ALG/TITCS L03-11

Greedy-choice of matroids

Let x be the largest element in S . We choose x if and only if $I \cup \{x\}$ is independent. We want to be sure that this is a safe choice.

Case 1:

Suppose we don't choose x . Question: is there an optimal solution that **does not** include x ?

It turns out that not only there is an optimal solution that does not include x , but **every solution does not include x** :

For any optimal solution I' of the subproblem, $I_0 \cup I'$ is independent.

If $x \in I'$, it implies $I_0 \cup \{x\} \subseteq I_0 \cup I'$ is independent due to the hereditary property, so we should have chosen x .

Therefore, $x \notin I'$ for any solution I' , and thus our choice not to include x is safe.

ALG/TITCS L03-12

Greedy-choice of matroids (cont'd)

Case 2:

If we do choose x , we want to find an optimal solution that include x .

Let $I' \subseteq S'$ be an optimal solution. If $x \in I'$, we are done.

Otherwise, we want to construct an optimal solution J' from I' such that $x \in J'$. We do it as follows.

Start with $J' = \{x\}$. Due to the exchange property, if $|J'| < |I'|$, there is an element $z \in I'$ which can be added to J' while keeping J' independent. We add z to J' .

This process is repeated until $|J'| = |I'|$. At the end, we basically deleted one element of I' and added back one element x .

Since x is an element with the largest weight in S' , the total weight of J' is at least that of I' , so J' is a set that we want.

ALG/TITCS L03-13

Optimal substructure of matroids

Now we know the first decision is not bad. We then change (reduce) the problem to another.

If the element x is not chosen, we change the problem by deleting x from S' . This is clearly a valid reduction: an optimal solution of the new problem is also an optimal solution of the original problem.

How about when x is really chosen?

In the algorithm, we change the problem by deleting x from S' and adding x to I_0 . Then we find an optimal solution I'' to the new problem, and say $I'' \cup \{x\}$ is actually an optimal solution of the old problem. Is this reduction valid?

ALG/TITCS L03-14

Optimal substructure of matroids (cont'd)

Assume that $I' = I'' \cup \{x\}$ is not optimal.

Let J' be an optimal solution of the original problem that include x . Since we have the greedy-choice property, such solution must exist.

Thus we have total weight of I' is strictly less than the total weight of J' .

Delete x from J' to form a new set J'' . Clearly, the total weight of J'' is strictly less than the total weight of J' . Therefore, if J'' is a solution to the reduced problem, we have a contradiction.

Now look at the problem again. We want a set $X \subseteq S' - \{x\}$ that makes $I_0 \cup \{x\} \cup X$ independent. Let's check whether J'' is such a set.

First, $J' \subseteq S'$ (as it is a solution to the original problem), so $J'' = J' - \{x\} \subseteq S' - \{x\}$.

Second, $I_0 \cup J'$ is independent (again, as J' is a solution to the original problem), meaning exactly that $I_0 \cup \{x\} \cup J''$ is independent. We arrive at a contradiction.

ALG/TITCS L03-15

Example: MST as an application of matroid

The MST problem can be viewed as an application of the matroid.

To apply the algorithms and proofs of matroid, we must formulate MST as a weighted independent subset problem of a matroid.

Since edges are the thing that is weighted, it is natural to use the edges as the elements of the matroid.

We want independent to eventually mean a tree, but that would not be hereditary. So we need a looser definition.

ALG/TITCS L03-16

The graph matroid

We say a set of edges are "independent" if it "can form part of a tree".

Or, in more simple terms, a set of edges are independent if they form no cycle, i.e. they form a forest.

Mathematically, S is the set of edges, and \mathcal{I} is the set of subsets of S that forms a forest.

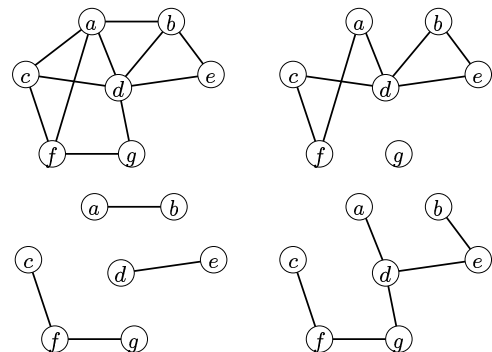
It turns out that (S, \mathcal{I}) is a matroid (the "graph matroid").

We are thus actually solving the "maximum weight spanning forest" problem, but of course a maximum weight spanning forest must be a tree, and thus a maximum spanning tree.

Since it is now a weighted maximum independent subset problem on a matroid, it can be solved using the algorithm that serve every other matroid.

ALG/TITCS L03-17

Example independent and dependent sets



ALG/TITCS L03-18

Spanning Forests form a matroid

Suppose we have a non-empty graph $G = (V, E)$. We want to see $(S, \mathcal{I}) = (E, \text{set of spanning forests of } G)$ is a matroid.

The non-empty and finite properties are satisfied: E is non-empty (given), and \emptyset is always a spanning forest (as it forms no cycle).

Spanning forest is hereditary: it is also trivial. A spanning forest is a set of edges that does not form a cycle, so any subset of a spanning forest cannot form a cycle. I.e. if $F \in \mathcal{I}$, $F' \in \mathcal{I}$ for all $F' \subseteq F$.

The most tricky property of a matroid is nearly always the exchange property.

ALG/TITCS L03-19

Spanning Forests form a matroid (cont'd)

Suppose we have two different spanning forests F_1 and F_2 of G , with $|F_1| < |F_2|$. We want to find an edge x in F_2 such that $F_1 \cup x$ is still a forest.

F_1 consists of some trees, and if we add an edge connecting two vertices of the same tree, we will get a cycle, and the tree is no longer a forest.

On the other hand, if we add an edge connecting two vertices of two different trees, we will merge the two trees and form a bigger tree, leaving the set of edges a forest.

Therefore, the key to choose a correct $x \in F_2$ is to choose one which connects two vertices of different trees in F_1 .

ALG/TITCS L03-20

Spanning Forests form a matroid (cont'd)

The question is, is such edge always exist?

Suppose not. Then all the edges of F_2 must connect vertices within a tree in F_1 .

Therefore, for each of the trees of F_1 of k vertices, F_2 contains at most $k - 1$ edges (the maximum number of edge without generating a cycle in the tree).

But F_1 also has $k - 1$ edges in each tree, i.e. F_2 cannot have more edges than $F_1 \Rightarrow$ Contradiction!

Therefore, there is always such an edge.

We thus shows the spanning forests of G form a matroid. Note that the above is only needed to establish that it is really a matroid. To solve the MST problem, we use...

ALG/TITCS L03-21

The greedy algorithm

```
Sort  $E$  in decending order of weights.  
 $I_0 \leftarrow \emptyset$   
for each  $e \in E$  (from the largest weight to the smallest)  
do  
  If  $I_0 \cup \{e\}$  is a forest  
  Then  
     $I_0 \leftarrow I_0 \cup \{e\}$   
Return  $I_0$ 
```

ALG/TITCS L03-22

Class announcement

Due to room arrangement of the JUPAS open-day, the lecture of Oct 14 will start at 2:15 pm.

Preview

Expect an assignment on the next week (Oct 7), to be due on Oct 21 before lecture.

ALG/TITCS L03-23