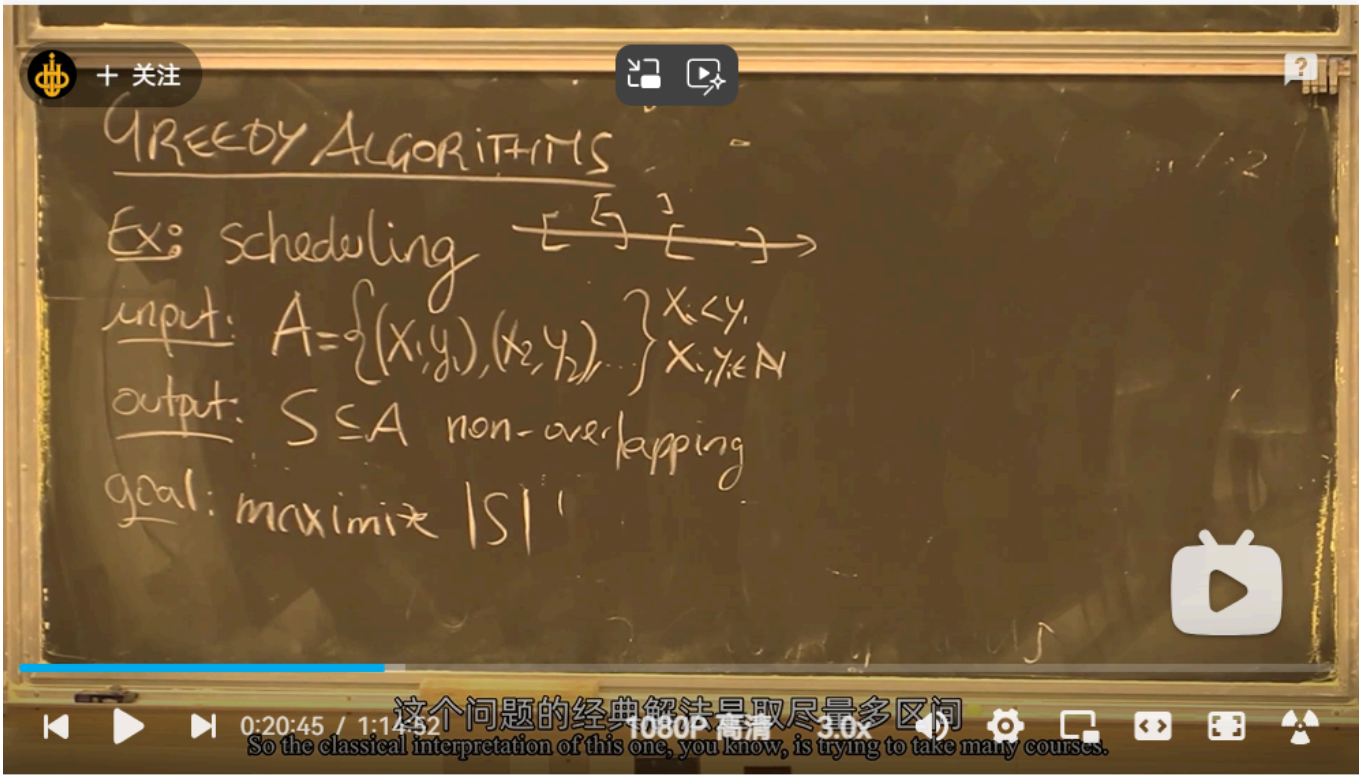
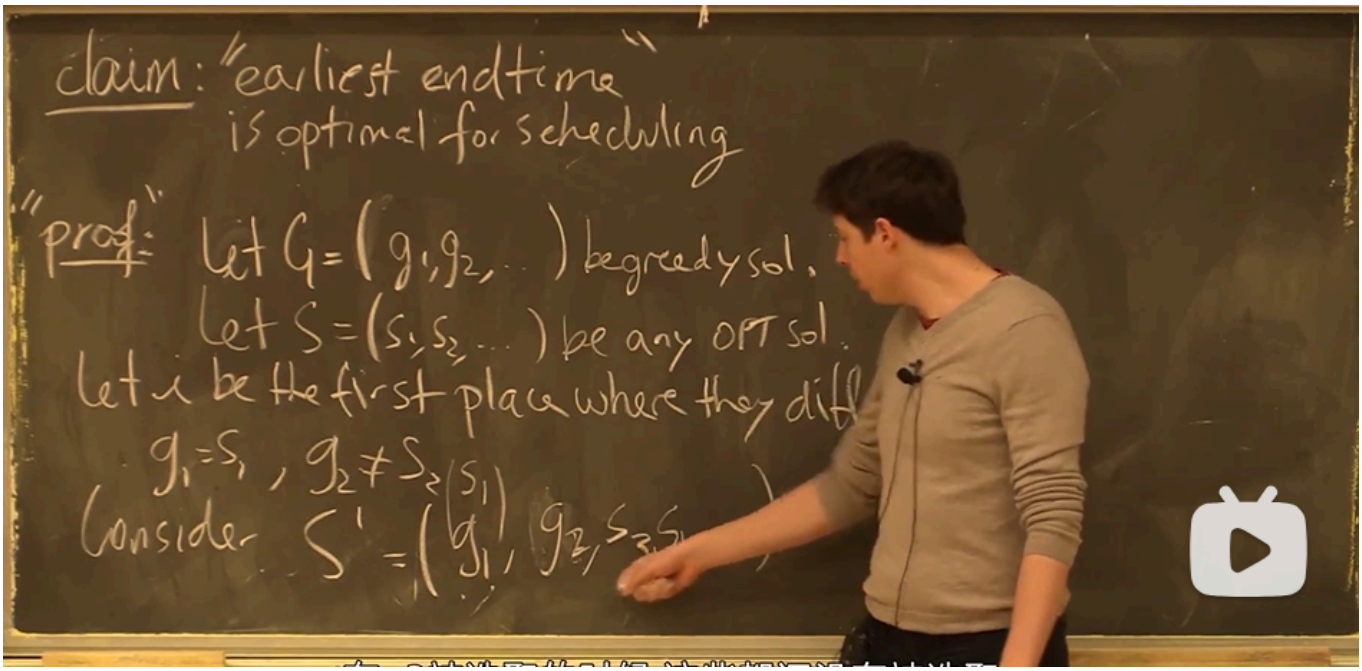
Greedy Algorithms

LOOK:In high level,there is no definition of greedy algorithm. It is more of a solution technique.

Schedule of a set of intervals that are non-overlapping in a line



way: just sort by endpoint time and pick the earliest not chosen one.(选择最早的未被选择的。)
proof:The main idea is that gn will be chosen before sn.(S is a set of optimal set.)



Set Cover

```
#include<iostream>
#include<vector>
#include<unordered_set>
using namespace std;

//函数用于找到集合覆盖
vector<int> setCover(vector<unordered_set<int>>
&sets,unordered_set<int>&universe)
{
  vector<int> result;
  unordered_set<int> covered;

  while(covered!=universe)
  {
      int bestSet = -1;
      int maxElement = 0;
      for(int i=0;i<sets.size();++i)
      {
          unordered_set<int> temp;
          for(int element:sets[i])
          {
              if(universe.count(element)&&!covered.count(element))
              {
```

```
                temp.insert(element);
            }
        }
        if(temp.size()>maxElement)
        {
            maxElement=temp.size();
            bestSet=i;
        }
    }

    if(bestSet==-1)
    {
        break;
    }

    result.push_back(bestSet);
    for(int element:sets[bestSet])
    {
        covered.insert(element);
    }
  }

  return result;
}


int main()
{
  vector<unordered_set<int>> sets = {
      {1, 2, 3},
      {2, 4},
      {3, 4},
      {4, 5}
  };


  unordered_set<int> universe = {1, 2, 3, 4, 5};
  vector<int> result = setCover(sets, universe);
  cout << "集合覆盖的集合索引：";
  for (int i : result)
```

```
    {
        cout << i << " ";
    }
    cout << endl;

    return 0;
}
```

MST

Kruskal Algorithm

Graph > kruskal Algorithm

Prim Algorithm

Graph > Prim Algorithm

Set_cover

We have a universe set{1,2,3,4,5}.
We also have some small sets:
{1, 2, 3},
{2, 4},
{3, 4},
{4, 5}
We need choose the least number of small sets to cover the universe set.

```
#include<iostream>
#include<vector>
#include<unordered_set>
using namespace std;

//函数用于找到集合覆盖
vector<int> setCover(vector<unordered_set<int>>
&sets,unordered_set<int>&universe)
{
    vector<int> result;
    unordered_set<int> covered;

    while(covered!=universe)
```

```
    {
        int bestSet = -1;
        int maxElement = 0;
        for(int i=0;i<sets.size();++i)
        {
            unordered_set<int> temp;
            for(int element:sets[i])
            {
                if(universe.count(element)&&!covered.count(element))
                {
                    temp.insert(element);
                }
            }
            if(temp.size()>maxElement)
            {
                maxElement=temp.size();
                bestSet=i;
            }
        }

        if(bestSet==-1)
        {
            break;
        }

        result.push_back(bestSet);
        for(int element:sets[bestSet])
        {
            covered.insert(element);
        }
    }

    return result;
}


int main()
{
    vector<unordered_set<int>> sets = {
        {1, 2, 3},
```

```
        {2, 4},
        {3, 4},
        {4, 5}
    };


    unordered_set<int> universe = {1, 2, 3, 4, 5};
    vector<int> result = setCover(sets, universe);
    cout << "集合覆盖的集合索引：";
    for (int i : result)
    {
        cout << i << " ";
    }
    cout << endl;

    return 0;
}
```