

Assignment Project Exam Help

Algorithms: https://powwapeler.com

Add Wechat powcoder

School of Computer Science and Engineering University of New South Wales

6. THE GREEDY METHOD

Activity selection problem.

Instance: A list of activities a_i , $(1 \le i \le n)$ with starting times s_i and finishing

ASSISHMENT PROJECT Exam Help
Task: Fine a maximum size subset of compatible activities.



Attempt Add twe Chat powcoder previously Add twe Chat powcoder



• The above figure shows this does not work...

Activity selection problem.

Instance: A list of activities a_i , $(1 \le i \le n)$ with starting times s_i and finishing

times f. No two activities can take place simultaneous Exam Help ASSIGNMENT PROJECT EXTENSE: Find a maximum size subset of compatible activities



Attempt 1 alvay choose the shoutest activity which does not conflict with the previously chosen activities remove the safectivity which does not conflict with the previously chosen activities are the safety of th



• The above figure shows this does not work...

Activity selection problem.

Instance: A list of activities a_i , $(1 \le i \le n)$ with starting times s_i and finishing times f_i . No two activities can take place simultaneously.

ASSIGNMENT PROJECT Exam Help

Task: Find a maximum size subset of compatible activities.



Attempt 1 alvay choose the shoutest activity which does not conflict with the previously chosen activities remote the saffecti glact www.sanl.dp.st?



• The above figure shows this does not work... (chosen activities in green, conflicting in red)

Activity selection problem.

Anstance: A list of activities a_i , D so in with stating times a_i and finishing a_i , a_i and a_i the property a_i and a_i the property a_i and a_i are a_i are a_i and a_i are a_i are a_i and a_i are a_i and a_i are a_i and a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i and a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i

Task: Find a maximum size subset of compatible activities.

• Attempt 2 Maybe we shall aways those a faction light conflicts with the fewest possible number of the remaining activities. It may appear that in this way we minimally restrict our next choice....

Add WeChat poweoder

ullet As appealing this idea is, the above figure shows this again does not work ...

Activity selection problem.

Anstance: A list of activities a_i , D so in with stating times a_i and finishing a_i , a_i and a_i the property a_i and a_i the property a_i and a_i are a_i are a_i and a_i are a_i are a_i and a_i are a_i and a_i are a_i and a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i are a_i are a_i are a_i and a_i are a_i and a_i are a_i are a_i are a_i are a_i and a_i are a_i are a_i

Task: Find a maximum size subset of compatible activities.

• Attempt 2 Meybe we should aways those the factifit of the conflicts with the fewest possible number of the remaining activities. It may appear that in this way we minimally restrict our next choice....

Add WeChat poweoder

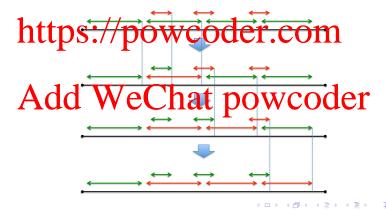
 \bullet As appealing this idea is, the above figure shows this again does not work \dots

The correct solution: Among the activities which do not conflict with the previously chosen activities always chose the one with the earliest end time.

Assignment Project Exam Help https://powcoder.com Add WeChat powcoder

- Find the first place where the chosen activity violates the greedy choice.

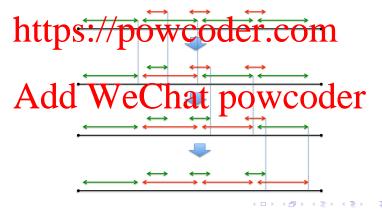
ssignment Project Exam Help



Show that any optimal solution can be transformed into the greedy solution with equal number of activities:

- Find the first place where the chosen activity violates the greedy choice.
- ssignment Project Exam Help

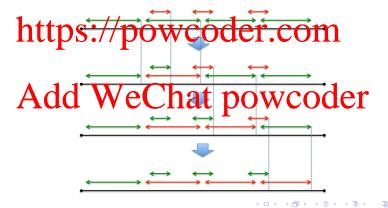
greedy solution, thus proving the greedy solution is also optimal.



Show that any optimal solution can be transformed into the greedy solution with equal number of activities:

- Find the first place where the chosen activity violates the greedy choice.
- Assignment Project Exam Help

greedy solution, thus proving the greedy solution is also optimal.



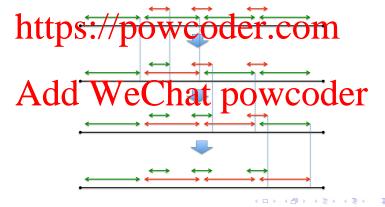
Show that any optimal solution can be transformed into the greedy solution with equal number of activities:

- Find the first place where the chosen activity violates the greedy choice.
- Show that replacing that activity with the greedy choice produces her spinging mich the smooth contact with an open contact with a cont

https://powcoder.com Add WeChat powcoder

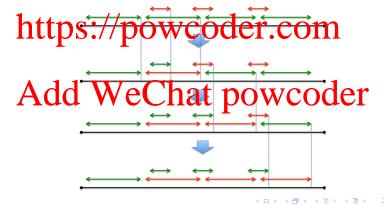
Show that any optimal solution can be transformed into the greedy solution with equal number of activities:

- Find the first place where the chosen activity violates.
 Show that replacing that activity with the greedy choice produces non the offact vitix and the contract of the co S Saft Grigs in it is manner till you "morph" your optimal solution in greedy solution, thus proving the greedy solution is also optimal.



Show that any optimal solution can be transformed into the greedy solution with equal number of activities:

- Find the first place where the chosen activity violates.
 Show that replacing that activity with the greedy choice produces non the offact vitix and the contract of the co S Saft Grigs in it is manner till you "morph" your optimal solution in greedy solution, thus proving the greedy solution is also optimal.



Assignment Project Exam Help

finishing times and sort them in an increasing order of their finishing time (the second coordinate).

- The https://powcoder.com
- We go through such a sorted list in order, looking for the first activity whose starting time is after the finishing time of the last taken activity
- Ever Add We Chat powcoder takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

• What is the time complexity of the algorithm?

ssignment Project Fxam Help finishing times and sort them in an increasing order of their finishing time (the second coordinate).

- https://powcoder.com
- We go through such a sorted list in order, looking for the first activity
 whose starting time is after the finishing time of the last taken activity
- Ever Add We Chat powcoder takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

• What is the time complexity of the algorithm?

ssignment Project Exam Help finishing times and sort them in an increasing order of their finishing time (the second coordinate).

- This https://powcoder.com
- We go through such a sorted list in order, looking for the first activity whose starting time is after the finishing time of the last taken activity
- Ever Add We Chat powcoder takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

• What is the time complexity of the algorithm?

ssignment Project Exam Help finishing times and sort them in an increasing order of their finishing time (the second coordinate).

- This https://powcoder.com
- We go through such a sorted list in order, looking for the first activity whose starting time is after the finishing time of the last taken activity.
- Ever Add We Chat powcoder takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

- What is the time complexity of the algorithm?
- Ssignment Project Exam Help finishing times and sort them in an increasing order of their finishing time (the second coordinate).
- This https://powcoder.com
- We go through such a sorted list in order, looking for the first activity whose starting time is after the finishing time of the last taken activity.
- Every actually is handle only back to properly for a general takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

- What is the time complexity of the algorithm?
- Ssignment Project Exam Help finishing times and sort them in an increasing order of their finishing time (the second coordinate).
- This https://powcoder.com
- We go through such a sorted list in order, looking for the first activity whose starting time is after the finishing time of the last taken activity.
- Every actually is handle only back to properly for a general takes O(n) time.
- Thus, the algorithm runs in total time $O(n \log n)$.

Activity selection problem II

A Sinstance: A list of activities a_i , $1 \le i \le n$ with starting time s_i and No two activities can take place simultaneously.

- Task: Find a subset of compatible activities of maximal total duration.
- solutions://powcoder.com
 to finding a selection with a largest number of non conflicting activities, i.e., the previous problem.
- Que Add a We Chat powcoder
- Greedy strategy no longer works we will need a more sophisticated technique.

Activity selection problem II

A Sinstance: A list of activities a_i , $(1 \le i \le n)$ with starting time s_i and high lims (1 + a) thu of activities are x_i described to a notivities can take place simultaneously.

• Task: Find a subset of compatible activities of maximal total duration.

• solutions://powcoder.com
to finding a selection with a largest number of non conflicting activities, i.e., the previous problem.

• QueAdd WeChat powcoder

 Greedy strategy no longer works - we will need a more sophisticated technique.

Activity selection problem II

A Single Hints $f = f_i + g$ thu is a fix its are Xf described in a fine so and No two activities can take place simultaneously.

- Task: Find a subset of compatible activities of maximal total duration.
- Solution: time all activities are of the same duration, this is equivalent to finding a selection with a largest number of non conflicting activities, i.e., the previous problem.
- QueAdd WeChat powcoder
- Greedy strategy no longer works we will need a more sophisticated technique.

Activity selection problem II

A Singling lines $G = \Pi_i + a$ thu is a fixed its are X_i the super X_i and No two activities can take place simultaneously.

- Task: Find a subset of compatible activities of maximal total duration.
- Solution: Indee all activities are of the same duration, this is equivalent to finding a selection with a largest number of non conflicting activities, i.e., the previous problem.
- Question? That the period that the latter of the safe
- Greedy strategy no longer works we will need a more sophisticated technique.

Activity selection problem II

A Single Hints $f = f_i + g$ thu is a fix its are Xf described in a fine so and No two activities can take place simultaneously.

- Task: Find a subset of compatible activities of maximal total duration.
- Solution: Indee all activities are of the same duration, this is equivalent to finding a selection with a largest number of non conflicting activities, i.e., the previous problem.
- Question? That Varpen if the attvition will of the safe
- Greedy strategy no longer works we will need a more sophisticated technique.

• Along the long, straight road from Loololong to Goolagong houses are scattered quite sparsely, sometimes with long gaps between two consecutive houses. Telstra must provide mobile phone service to people who live alongside the road, and the range of Telstra's cell base station is Skip O span appropriate for the minima name of base stations alongside the road, that is sufficient to cover all nouses.

**https://powcoder.com

Add WeChat powcoder

Once again, along the long, straight road from Loololong (on the West) to Goolagong (on the East) houses are scattered quite sparsely, sometimes with long gaps between two coased three houses. Te strain list a must be purely to be a looked plant terricely depleted by the literature library and the range of Telstras cell base station is 5km.

- One of Telstra's engineer started with the house closest to Loololong and put https://powcoder.com/nost house not https://powcoder.com/nost house the East of it and continued in this way till he reached Goolagong.
- Add WeChat powcoder
- Is there a placement of houses for which the associate is right?



- Once again, along the long, straight road from Loololong (on the West) to Goolagong (on the East) houses are scattered quite sparsely, semestimes with long gaps by tween two coased three houses. To star must be depute mobile plant terricely depute who it was regarded the range of Telstras cell base station is 5km.
- One of Telstra's engineer started with the house closest to Loololong and put and the part of the Hast. He therefore the performed the permost house not already in the range of the tower and praced another tower 5 km to the East of it and continued in this way till he reached Goolagong.
- Add WeChat powcoder
- Is there a placement of houses for which the associate is right?



- Once again, along the long, straight road from Loololong (on the West) to Goolagong (on the East) houses are scattered quite sparsely, semestimes with long gaps by tween two coased three houses. To star must be depute mobile plant terricely depute who it was regarded the range of Telstras cell base station is 5km.
- One of Telstra's engineer started with the house closest to Loololong and put a dwarf km ayay to the Fast. He therefound the yestmost house not areary in the range of the tower and placed another tower 5 km to the East of it and continued in this way till he reached Goolagong.
- His junior associated decaded the same but starting from the East and moving westwards and comes that his method we were fowers.
- Is there a placement of houses for which the associate is right?

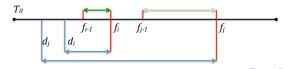


- Once again, along the long, straight road from Loololong (on the West) to Goolagong (on the East) houses are scattered quite sparsely, semestimes with long gaps by tween two coased three houses. To star must be depute mobile plant terricely depute who it was regarded the range of Telstras cell base station is 5km.
- One of Telstra's engineer started with the house closest to Loololong and put hower 5km away to the Hase. He therefound the yestmost house not already in the range of the tower and praced another tower 5 km to the East of it and continued in this way till he reached Goolagong.
- His junior associated did exactly the same but starting from the East and moving westwards and falmed that his method we will fowers.
- Is there a placement of houses for which the associate is right?



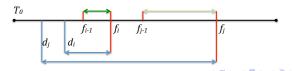
- Instance: A start time T_0 and a list of jobs a_i , $(1 \le i \le n)$, with duration times t_i and deadlines d_i . Only one job can be performed at any time all jobs f_i such that it has incurred lateness $f_i = f_i - d_i$.
 - Task: Schedule all the jobs so that the lateness of the job with the largest https://powcoder.com

 - Add WeChat powcoder



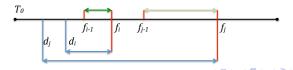
- Instance: A start time T_0 and a list of jobs a_i , $(1 \le i \le n)$, with duration times t_i and deadlines d_i . Only one job can be performed at any time all jobs See Proposition 1 a jet f is included at a fixed the f in the we see that it has incurred lateness $f = f_i - d_i$.
 - Task: Schedule all the jobs so that the lateness of the job with the largest https://powcoder.com

 - Add WeChat powcoder



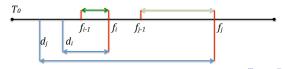
- Instance: A start time T_0 and a list of jobs a_i , $(1 \le i \le n)$, with duration times t_i and deadlines d_i . Only one job can be performed at any time all jobs See Proposition 1 a joint of at a fixed the first we see that it has incurred lateness $f = f_i - d_i$.
 - Task: Schedule all the jobs so that the lateness of the job with the largest ntips://powcoder.com

 Solution: Ignore job durations and schedule jobs in the increasing order of
 - deadlines.
 - Add WeChat powcoder



- Instance: A start time T_0 and a list of jobs a_i , $(1 \le i \le n)$, with duration times t_i and deadlines d_i . Only one job can be performed at any time all jobs Sve Q le double Ala jor (s) hopete at a box big the f > (we say that it has incurred lateness $U = f_i - d_i$.
 - Task: Schedule all the jobs so that the lateness of the job with the largest ntips://powcoder.com

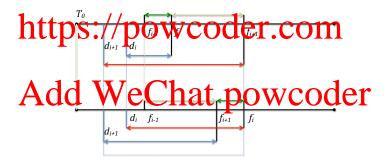
 Solution: Ignore job durations and schedule jobs in the increasing order of
 - deadlines.
 - Optimality: Consider any optimal solution. We say that jobs u_i and jobs a_j form an inversion if v_i of its schedular before of v_i of v_j of v_j



Minimising job lateness

Assignment Project Exam Help

adjacent jobs, eventually all the inversions will be eliminated.

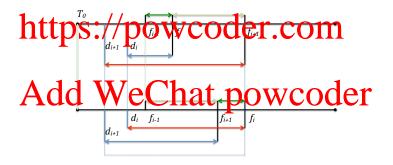


• Note that swapping adjacent inverted jobs reduces the larger lateness!

Minimising job lateness

• We will show that there exists a scheduling without inversions which is also

Second Republicant of the manage to eliminate all inversions between padjacent jobs, eventually all the inversions will be eliminated.

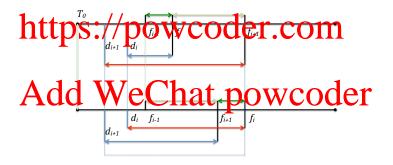


• Note that swapping adjacent inverted jobs reduces the larger lateness!

Minimising job lateness

• We will show that there exists a scheduling without inversions which is also

Second the Bubble of the five manage to channate all inversions between the adjacent jobs, eventually all the inversions will be eliminated.



Note that swapping adjacent inverted jobs reduces the larger lateness!

Tape storage

As a stage of the prices of the prices of the price of the prices of the beginning of the tape and scan it until the file is found and read.

- * https://powcoder.com
- Solution: If the files are stored in order $l_1, l_2, \dots l_n$, then the expected time is proportional to

Add WeChat powcoder

 $nl_1 + (n-1)l_2 + (n-2)l_3 + \ldots + 2l_{n-1} + l_n$

Tape storage

Sustance this Prince Prince which are perpending the beginning of the tape and scan it until the file is found and read.

- \bullet Task Order the files on the tape so that the average (expected) retrieval time is minimised DS://DOWCOGET.COM
- Solution: If the files are stored in order $l_1, l_2, \dots l_n$, then the expected time is proportional to

Add WeChat powcoder

 $nl_1 + (n-1)l_2 + (n-2)l_3 + \ldots + 2l_{n-1} + l_n$

Tape storage

- Sustance this Prince the Each of the lavey o
 - \bullet Task Order the files on the tape so that the average (expected) retrieval time is militarily by the content of the conten
 - Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

$$\begin{array}{c} \mathbf{Add_+WeChat.powcoder} \\ {}_{nl_1+(n-1)l_2+(n-2)l_3+\ldots+2l_{n-1}+l_n} \end{array}$$

Tape storage

Sustance this Prince the Each of the lavey o

- \bullet Task Order the files on the tape so that the average (expected) retrieval time is milines to ps./powcoder.com
- Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

Tape storage II

Instance: A list of n files \mathbf{j} of lengths l_i and probabilities to be needed p_i \mathbf{j} of $\mathbf{j$

• Task: Order the files on the tape so that the expected retrieval time is minihttps://powcoder.com

• Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

Add WeChat powcoder

Tape storage II

- Task: Order the files on the tape so that the expected retrieval time is minimised to minimised power in the minimised power in the expected retrieval time is minimised by the expected retrieval time in the expected retrieval time is minimised by the expected retrieval time is minimised by the expected retrieval time in the expected retrieval time is minimised by the expected retrieval time is minimised by the expected retrieval time in the expected retrieval time is minimised by the expected retrieval time in the expected retrieval time is minimised by the expected retri
- Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

Add WeChat powcoder

Tape storage II

A S $[1, p_i]$ F1 Vilit law to be stored on a tape. To extred a file, and and read.

- Task: Order the files on the tape so that the expected retrieval time is minibised to the minibised to the
- Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

Add-WeChat-powcoder)pn

Tape storage II

A S S 1 St Hi White lad to be stored of take. To arrive a file, and and read start from the beginning of the tape and scan it until the file is fund and read.

- Task: Order the files on the tape so that the expected retrieval time is minibised to the minibised to the composition of the
- Solution: If the files are stored in order $l_1, l_2, \ldots l_n$, then the expected time is proportional to

Add - We Chat - powcoder)pn

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

Assignment Project Exam Help

 $E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1}$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k \leq p_{k+1} / l_{k+1}$.
- ConsAdd > WeChat pow.coder that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

Assignment Project Exam Help

 $E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \dots + (l_1 + l_2 + l_3 + \dots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + https://powcoder.com + l_n) p_n$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k \leq p_{k+1} / l_{k+1}$.
- ConsAdd > WeChat pow.coder that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

 $E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + ... + (l_1 + l_2 + l_3 + ... + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_2 + l_3 + ... + l_{k-1} + l_n) p_n$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k \leq p_{k+1} / l_{k+1}$.
- ConsAdd > WeChat pow.coder that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

$\mathbf{A} \overset{E}{\underset{\mathrm{and}}{\mathbf{Signment}}} \overset{p_1}{\underset{\mathrm{roject}}{\mathbf{Exam}}} \overset{l_1}{\underset{\mathrm{log}}{\mathbf{Exam}}} \overset{l_2}{\underset{\mathrm{log}}{\mathbf{Help}}}$

$$E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_n$$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k \le p_{k+1} / l_{k+1}$.
- ConsAdd > WeChat pow.coder that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

$\mathbf{A} \overset{E}{\underset{\mathrm{and}}{\mathbf{Signment}}} \overset{p_1}{\underset{\mathrm{roject}}{\mathbf{Exam}}} \overset{l_1}{\underset{\mathrm{log}}{\mathbf{Exam}}} \overset{l_2}{\underset{\mathrm{log}}{\mathbf{Help}}}$

$$E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_n$$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k < p_{k+1} / l_{k+1}$.
- ConsAdd > We Chat powcoder that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

$\mathbf{A} \overset{E}{\underset{\mathrm{and}}{\mathbf{S1gnment}}} \overset{p_1}{\underset{\mathrm{roject}}{\mathbf{E2m}}} \overset{l_1}{\underset{\mathrm{roject}}{\mathbf{E2m}}} \overset{l_2}{\underset{\mathrm{h}}{\mathbf{Help}}} \overset{l_1}{\underset{\mathrm{h}}{\mathbf{E2p}}}$

$$E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_n$$

- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k < p_{k+1} / l_{k+1}$.
- Consequence > Wf Coordinate $p_k / l_k < p_{k-1} / l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1} / l_{k+1} has been put after a file f_k with a smaller ratio p_k / l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

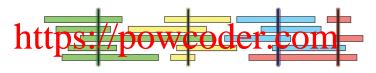
- Let us see what happens if we swap to adjacent files f_k and f_{k+1} .
- The expected time before the swap and after the swap are, respectively,

$\mathbf{A} \overset{E}{\underset{\text{and}}{\mathbf{Signment}}} \overset{l_{1}+l_{2}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{and}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+l_{3}+\ldots+l_{k}}{\underset{\text{pred}}{\mathbf{Pt}}} \overset{l_{1}+l_{2}+l_{3}+l_{$

$$E' = l_1 p_1 + (l_1 + l_2) p_2 + (l_1 + l_2 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_{k+1} + (l_1 + l_3) p_3 + \ldots + (l_1 + l_2 + l_3 + \ldots + l_{k-1} + l_{k+1}) p_n$$

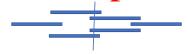
- Thus, $E E' = l_k p_{k+1} l_{k+1} p_k$, which is positive just in case $l_k p_{k+1} > l_{k+1} p_k$, i.e., if $p_k / l_k < p_{k+1} / l_{k+1}$.
- Consequence > Mf Edorbii $2/l_k$ Q_kQ/M_1 , Chio Gest that the swap decreases the expected time just in case $p_k/l_k < p_{k+1}/l_{k+1}$, i.e., if there is an inversion: a file f_{k+1} with a larger ratio p_{k+1}/l_{k+1} has been put after a file f_k with a smaller ratio p_k/l_k .
- For as long as there are inversions there will be inversions of consecutive files and swapping will reduce the expected time. Consequently, the optimal solution is the one with no inversions.

• Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P; see the figure below. Describe and analyse an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists S we arrais 17. Man X 19. Theoretical mig X hard figure.



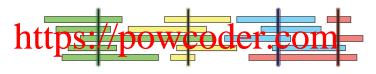
A set of intervals stabbed by four points (shown here as vertical segments)

• Add WeChat powcoder



- Hint: the interval which ends the earliest has to be stabbed
- What is the best place to stab it?

• Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P; see the figure below. Describe and analyse an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists S we are 1 1 1 and X 1 1 1 Teoresen ing X had fit and right end with of the intervals in X.



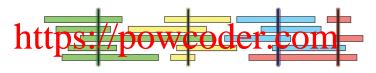
A set of intervals stabbed by four points (shown here as vertical segments)

• Is it Adda twaccapatospouwcodet?



- Hint: the interval which ends the earliest has to be stabbed
- What is the best place to stab it?

• Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P; see the figure below. Describe and analyse an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists S we are 1 1 1 and X 1 1 1 Teoresen ing Xh eff and right endrouts of the intervals in X.



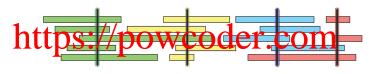
A set of intervals stabbed by four points (shown here as vertical segments)

• Is it Adda twaccapatospouwcodel?



- Hint: the interval which ends the earliest has to be stabbed.
- What is the best place to stab it?

• Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P; see the figure below. Describe and analyse an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists S we are 1 1 1 and X 1 1 Teoresen mg X had fit and right end wants of the intervals in X.



A set of intervals stabbed by four points (shown here as vertical segments)

• Is it Adda twaccateatospouwcodet?



- Hint: the interval which ends the earliest has to be stabbed.
- What is the best place to stab it?

0-1 knapsack problem

A Sand an Anna Long It limit W of works lock. Exam Help

• Task: Find a subset S of all items available such that its weight does not

https://powcoder.com

• Assume there are just three items with weights and values: (10kg, \$60), (20kg, 100) (20kg, 100) a Chapsack of capacity W = 50kg.

Add WeChat powcoder

Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution

- So when does the Greedy Strategy work??

0-1 knapsack problem

A Sand an kink the filling with the best of the sand values we for the sand and kink the filling with the best of the sand and was the sand the san

• Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.

https://powcoder.com

• Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100) (20kg, \$100) a Chapsack of capacity W = 50kg.

Add WeChat powcoder

Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution

- So when does the Greedy Strategy work??

0-1 knapsack problem

Assignment of weights mand values v_i for iscrete items at $i \in I$ p

- Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.
- Can we always choose the property of the mightest value per unit weight?
- Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100) (30kg) a Chapsack of capacity W = 50kg. The standard powcoder
- Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution is to take (20kg, \$100) and (30kg, \$120)!
- So when does the Greedy Strategy work??
- Unfortunately there is no easy rule...



0-1 knapsack problem

A Sad galand wight mit w by the costck. Exam Help

- Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.
- Can we always choose the item with the nightest value per unit weight?
- Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100), (30kg \$120) and 4 knapsack of capacity W = 50kg.
- Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution is to take (20kg, \$100) and (30kg, \$120)!
- So when does the Greedy Strategy work??
- Unfortunately there is no easy rule...



0-1 knapsack problem

A Sand an Aright limit W by the Orstick Exam Help

- Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.
- Can we alway shoose the pitch with the nightest value per unit weight?
- Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100), (30kg \$120) and 4 knapsack of capacity W = 50kg.
- Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution is to take (20kg, \$100) and (30kg, \$120)!
- So when does the Greedy Strategy work??
- Unfortunately there is no easy rule...

0-1 knapsack problem

A Sand an Aright limit W by the Orstick Exam Help

- Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.
- Can we always choose the item with the nightest value per unit weight?
- Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100), (30kg \$120) and 4 knapsack of capacity W = 50kg.
- Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution is to take (20kg, \$100) and (30kg, \$120)!
- So when does the Greedy Strategy work??
- Unfortunately there is no easy rule...



0-1 knapsack problem

A Sand an Aright limit W by the Orstick Exam Help

- Task: Find a subset S of all items available such that its weight does not exceed W and its value is maximal.
- Can we alway shoose to prove the nightest value per unit weight?
- Assume there are just three items with weights and values: (10kg, \$60), (20kg, \$100), (30kg, \$120) and 4 knapsack of capacity W=50kg.
- Greedy would choose (10kg, \$60) and (20kg, \$100), while the optimal solution is to take (20kg, \$100) and (30kg, \$120)!
- So when does the Greedy Strategy work??
- Unfortunately there is no easy rule...

- You are allowed to merge any two arrays into a single new sorted array and proceed in this manner until only one array is left.
 - Designation by Designation Des
- This Add s We Chathpowcoder Greedy method, which is, urgually, along the greedy method!

- You are allowed to merge any two arrays into a single new sorted array and proceed in this manner until only one array is left.
- Desi https://powcoder.comal total number of moves of elements of the arrays and give an informal justification why your algorithm is optimal.
- This Add s We Chathpowcoder Greedy method, which is, urguely, along the greedy method!

- You are allowed to merge any two arrays into a single new sorted array and proceed in this manner until only one array is left.
- Design that the Cheve Chief and the Commandation of moves of elements of the arrays and give an informal justification why your algorithm is optimal.
- This Add s We Chath powcoder Greedy method, which is, arguently, along the greedy method!

- You are allowed to merge any two arrays into a single new sorted array and proceed in this manner until only one array is left.
- Design that the chief the chief that the common total number of moves of elements of the arrays and give an informal justification why your algorithm is optimal.
- This problem is somewhat related to the ax unplication of the Greedy method, which is, arguably, among the most important applications of the greedy method!

• Assume you are given a set of symbols, for example the English alphabet plus

- One way of doing so is to reserve bit strings of equal and sufficient length, https://powcoder.com^{25 = 32}
- bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- Add WeChat powcoder
- However, if the codes are of variable length, then how can we partition a
- One way of insuring unique readability of codes from a single bitstream is to
- Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus purctuation marks and a blank space (to be used between words). SI and the sembles to see the sembles of the sembles of the sembles can be decoded in an mambiguous way.
 - One way of doing so is to reserve bit strings of equal and sufficient length, https://powcoder.com^{25 = 32}
 - bits and use a lookup table to decode the text.
 - However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
 - Add WeChat powcoder
 - However, if the codes are of variable length, then how can we partition a
 - One way of insuring unique readability of codes from a single bitstream is to
 - Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus punctuation marks and a black space (to be used between words). It is a such symbols can be decoded in an anambiguous way.
 - One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus is printing in symbols, you would need strings of length $5 \cdot (2^5 = 32)$.
 - To diffuse of the power of 5 bits and use a lookup table to decode the text.
 - However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
 - One Addra We Chat powcoder, 'i' or longer codes.
 - However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
 - One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
 - Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).

 Solvent and the plus purctuation marks and a brank space (to be used between words).
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus benefit properly by properly and properly symbols, you would need strings of length $5 \cdot (2^5 = 32)$.
- To decide a piece of tex you will death the trabit Cresh into groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One Addr We Chat powcoder or longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus purictuation marks and a brank space (to be used between words).

 Solvent in the symbol of the symbols of the property strings at that sequences of possible to the symbols can be decoded in an anambiguous way.
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus by nettration symbols, you would need strings of length $5 (2^5 = 32)$.
- To decide a piece of tex you will dealth by the bit Cresh into groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One Addra We Chat powcoder ', 'i' or longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus purictuation marks and a brank space (to be used between words).

 Solvent in the symbol of the symbols of the property strings at that sequences of possible to the symbols can be decoded in an anambiguous way.
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus to protect that one symbols, you would need strings of length $5 (2^5 = 32)$.
- To decide a piece of tex you would partitled the bit the hint groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One would refer at encoung in which frequent symbols such as (2) fe', 'i' or 't' have short codes while infrequent ones, such as w','x' and y' can have longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus purictuation marks and a brank space (to be used between words).

 Solvent in the symbol of the symbols of the property strings at that sequences of possible to the symbols can be decoded in an anambiguous way.
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus benefit problems by models, you would need strings of length 5 ($2^5 = 32$).
- To decide a piece of tex you will death the trabit Cresh into groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One would refer a specifig in which frequent ones, such as w,'x and y can have longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called the prefix codes.

- Assume you are given a set of symbols, for example the English alphabet plus punctuation marks and a black space (to be used between words).

 Solvandence there symbols be that sequences of particles of particles
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus benefit problems by models, you would need strings of length 5 ($2^5 = 32$).
- To decide a piece of tex you would partitled the bit Care in into groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One would refer a specifig in which frequent ones, such as w,'x and y can have longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called *the prefix codes*.

- Assume you are given a set of symbols, for example the English alphabet plus punctuation marks and a black space (to be used between words).

 Solvandence there symbols be that sequences of particles of particles
- One way of doing so is to reserve bit strings of equal and sufficient length, given the number of distinct symbols to be encoded; say if you have 26 letters plus benefit problems by models, you would need strings of length 5 ($2^5 = 32$).
- To decide a piece of tex you would partitled the bit Care in into groups of 5 bits and use a lookup table to decode the text.
- However this is not an economical way: all the symbols have codes of equal length but the symbols are not equally frequent.
- One would refer a knowing in which frequent ones, such as w','x and y can have longer codes.
- However, if the codes are of variable length, then how can we partition a bitstream UNIQUELY into segments each corresponding to a code?
- One way of insuring unique readability of codes from a single bitstream is to ensure that no code of a symbol is a prefix of a code for another symbol.
- Codes with such property are called the prefix codes.

Assignment Project Exam Help

• We can now formulate the problem:

Given the frequencies (probabilities of occurrences) of each symbol, design an opting trifficedes itel profit rode such that the expected length of an encoded test is small possible. Colder the expected length of an

- Note that this amounts to saying that the average number of bits per symbol in an "average" text is as small as possible.
- We now sketch the algorithm informally; please see the textbook for details and the proof of optimality.

As singular tentrs or tradictory to the (x, y) coordinates of each tower and its radius of range. When a tower is activated, all towers within the radius of range of the tower will also activate, and those can cause other towers to activate and so on.

You hear the equip son of these towers with seismic sensors so that when these sensors activate the towers where these sensors are located all towers will eventually get activated and send a tsunami warning.

• The Addie We Chatipowcoder of towers you must equip with seismic sen processing of the control of the control

Assignments of rode $\mathbf{C}_{\mathbf{g}}$ tsuch a tower is activated, all towers within the radius of range of the tower will also activate, and those can cause other towers to activate and so on.

- You need to equip some of these towers with seismic sensors so that when these sensors activate the towers where these sensors are located all towers will eventually get activated and send a tsunami warning.
- The Addle We Chatipowcoder of towers you must equip with seismic sen pro-

- Assignment Project Exams. Help given the (x, y) coordinates of each tower and its radius of range. When a tower is activated, all towers within the radius of range of the tower will also activate, and those can cause other towers to activate and so on.
 - You need to equip some of these towers with seismic sensors so that when these sensors activate the towers where these sensors are located all towers will eventually get activated and send a tsunami warning.
 - The sol is to design are gori in which to towers you must equip with seismic sensors.

• Someone has proposed the following two algorithms:

Assignment Project Exam Help

https://powcoder.com

- Add WeChat powcoder 2
- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Someone has proposed the following two algorithms:

Assi Algorithm 1: Find the unactivated tower with the largest radius (if Sulful Active the Law & den lactive t

- $\begin{array}{c} \text{Algorithm 2: Find the unactivated twer with the largest number of } \\ \text{POWCOder: Com} \\ \text{ftmost tower.} \end{array}$
- * Add WeChat powcoder *
- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Someone has proposed the following two algorithms:

Assi Algorithm 1: Find the unactivated tower with the largest radius (if Sulful Assirable same lating Gic the any X dentil Activated towers. Find and remove all activated towers. Repeat.

Algorithm 2: Find the unactivated tower with the largest number of the swifting to the largest number of the swifting that the swifting th

* Add WeChat powcoder 2

- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Someone has proposed the following two algorithms:

Assi Algorithm 1: Find the unactivated tower with the largest radius (if Sultinate Helsane lating Gic the any X denni Activated tower. Find and remove all activated towers. Repeat.

- Algorithm 2: Find the unactivated tower with the largest number of the swifting to the largest number of the swifting that the swifting th
- Give examples which show that neither Algorithm 1 nor Algorithm 2 solve the puber wrett. That powcoder
- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Someone has proposed the following two algorithms:

Assi Algorithm 1: Find the unactivated tower with the largest radius (if Sultinate Helsane lating Gic the any X denni Activated tower. Find and remove all activated towers. Repeat.

- Algorithm 2: Find the unactivated tower with the largest number of live is which it also where it is the first tower. Repeat.
- Give examples which show that neither Algorithm 1 nor Algorithm 2 solve the puber wrett. That powcoder
- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Someone has proposed the following two algorithms:

Assi Algorithm 1: Find the unactivated tower with the largest radius (if Sulful Assi Helsane lating GC the any X den HA divact tower. Find and remove all activated towers. Repeat.

- Algorithm 2: Find the unactivated tower with the largest number of live is which it also where it is the first tower. Repeat.
- Give examples which show that neither Algorithm 1 nor Algorithm 2 solve the puber wrett. That powcoder
- Design an algorithm which correctly solves the problem.
- Solving this problem involves several important concepts which we now revisit.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from \underline{v} to u and a path from u to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction direction.
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Fin Add WeChat powcoder from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from v to v and a path from v to v.

Assignment Project Exam Help

• Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction direction.

- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Fin Add WeChat powcoder from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from \underline{v} to u and a path from u to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V but with the set of edges E_{rev} obtained by reversing the direction of alges/F0. WCOCH. COM
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Fin Add WeChat powcoder from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from \underline{v} to u and a path from u to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction of a Science V by V coder. Com
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Fin Add WeChat powcoder from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from v to v and a path from v to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction of a Science V by V coder. Com
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Find and det We Chat power oder, from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from v to v and a path from v to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction of a Science V by V coder. Com
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Find and det We Chat power oder, from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Given a directed graph G = (V, E) and a vertex v, the strongly connected component of G containing v consists of all vertices $u \in V$ such that there is a path in G from v to v and a path from v to v.

- Construct another graph $G_{rev} = (V, E_{rev})$ consisting of the same set of vertices V, but with the set of edges E_{rev} obtained by reversing the direction of a Science V by V coder. Com
- Use BFS to find the set $D \subseteq V$ of all vertices in V which are reachable in G from v.
- Find and det We Chat value of the from v.
- The strongly connected component C of G containing v is just the set $C = D \cap R$.
- Clearly, distinct strongly connected components are disjoint sets.

• Let S_G be the set of all strongly connected components of a graph G.

Assignment Project Exam Help between the strongly connected components so that if $\sigma_1 \in S_G$ and

between the strongly connected components so that if $\sigma_1 \in S_G$ and $\sigma_2 \in S_G$ and $\sigma_1 \neq \sigma_2$ then there exists an edge from σ_1 to σ_2 in E^* just in case there exist a vertex $u_1 \in \sigma_1$ and a vertex $u_2 \in \sigma_2$ so that there exist a vertex $u_1 \in \sigma_1$ and a vertex $u_2 \in \sigma_2$ so that there

https://powcoder.com

- Clearly, $\Sigma = (S_G, E^*)$ is a directed acyclic graph, and thus allows a topological sorting of vertices.
- Top-Aiddt We Chaterpowcoder (enumeration) of its vertices $\sigma_i \in S_G$ sup that if there exists an edge $(\sigma_i, \sigma_j) \in E^*$ then vertex σ_i precedes σ_j in such an ordering , i.e., i < j
- How do we topologically sort a directed acyclic graphs?

• Let S_G be the set of all strongly connected components of a graph G.

Assignment (S_C, F^*) with the expectation S_G and S_G and a vertex S_G and S_G

• Clearly, $\Sigma = (S_G, E^*)$ is a directed acyclic graph, and thus allows a topological sorting of vertices.

• Top Add We Chat powcoder (enumeration) of its vertices $\sigma_i \in S_G$ sup that if there exists an edge $(\sigma_i, \sigma_j) \in E^*$ then vertex σ_i precedes σ_j in such an ordering , i.e., i < j.

• How do we topologically sort a directed acyclic graphs?

• Let S_G be the set of all strongly connected components of a graph G.

Assignment (F, F) between the strongly connected components so that if $\sigma_1 \in S_G$ and $\sigma_2 \in S_G$ and $\sigma_1 \neq \sigma_2$ then there exists an edge from σ_1 to σ_2 in E^* just in case there exist a vertex $u_1 \in \sigma_1$ and a vertex $u_2 \in \sigma_2$ so that there exist an edge from σ_1 to σ_2 in σ_2 in σ_2 in σ_3 to σ_4 and a vertex σ_4 so that there exist an edge from σ_4 to σ_4 in σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist and edge from σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist and edge from σ_4 to σ_4 and σ_4 are exist an edge from σ_4 to σ_4 and σ_4 are exist an edge from σ_4 to σ_4 and σ_4 are exist an edge from σ_4 to σ_4 and σ_4 are exist an edge from σ_4 to σ_4 and σ_4 are exist an edge from σ_4 are exist an edge from σ_4 and σ_4 are exist an edge from

- Clearly, $\Sigma = (S_G, E^*)$ is a directed acyclic graph, and thus allows a topological sorting of vertices.
- Top-Aidd twe Chat poweoder (enumeration) of its vertices $\sigma_i \in S_G$ sup that if there exists an edge $(\sigma_i, \sigma_j) \in E^*$ then vertex σ_i precedes σ_j in such an ordering , i.e., i < j.
- How do we topologically sort a directed acyclic graphs?

• Let S_G be the set of all strongly connected components of a graph G.

Assignment (S, F) between the strongly connected components so that if $\sigma_1 \in S_G$ and $\sigma_2 \in S_G$ and $\sigma_1 \neq \sigma_2$ then there exists an edge from σ_1 to σ_2 in E^* just in case there exist a vertex $u_1 \in \sigma_1$ and a vertex $u_2 \in \sigma_2$ so that there exist an edge from σ_1 to σ_2 in σ_2 in σ_2 in σ_3 in the context σ_4 and σ_4 vertex σ_4 in σ_4 and σ_4 vertex σ_4 in σ_4 in

- Clearly, $\Sigma = (S_G, E^*)$ is a directed acyclic graph, and thus allows a topological sorting of vertices.
- Topologica of topological experiments of its vertices $\sigma_i \in S_G$ such that if there exists an edge $(\sigma_i, \sigma_j) \in E^*$ then vertex σ_i precedes σ_j in such an ordering , i.e., i < j.
- How do we topologically sort a directed acyclic graphs?

• Let S_G be the set of all strongly connected components of a graph G.

Assignment (F, F) between the strongly connected components so that if $\sigma_1 \in S_G$ and $\sigma_2 \in S_G$ and $\sigma_1 \neq \sigma_2$ then there exists an edge from σ_1 to σ_2 in E^* just in case there exist a vertex $u_1 \in \sigma_1$ and a vertex $u_2 \in \sigma_2$ so that there exist an edge from σ_1 to σ_2 in σ_2 in σ_2 in σ_3 in σ_4 and a vertex σ_4 so that there exist an edge from σ_4 to σ_4 in σ_4 and σ_4 in σ_4 so that there exist an edge from σ_4 to σ_4 in σ_4 and σ_4 in σ_4 and σ_4 in σ_4

- Clearly, $\Sigma = (S_G, E^*)$ is a directed acyclic graph, and thus allows a topological sorting of vertices.
- Topologica of topological educations of its vertices $\sigma_i \in S_G$ such that if there exists an edge $(\sigma_i, \sigma_j) \in E^*$ then vertex σ_i precedes σ_j in such an ordering , i.e., i < j.
- How do we topologically sort a directed acyclic graphs?

- ullet L \leftarrow Empty list that will contain ordered elements
- $S \leftarrow$ Set of all nodes with no incoming edge

Assignment Project Exam Help

https://powcoder.com

ullet remove edge e from the graph;

Add WeChat powcoder

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

Now it should be easy to use the Greedy Strategy to solve the problem of finding the fewest number of towers which you must equip with seismic sensors, so that all emergency transmission towers are activated.

- $L \leftarrow$ Empty list that will contain ordered elements
- \bullet $S \leftarrow \mathbf{Set}$ of all nodes with no incoming edge

Assignment Project Exam Help

https://powcoder.com
for each node v to tail of v:

powcoder.com
the remove edge e from the graph:

Add WeChat powcoder

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

Now it should be easy to use the Greedy Strategy to solve the problem of finding the fewest number of towers which you must equip with seismic sensors, so that all emergency transmission towers are activated.

- $L \leftarrow$ Empty list that will contain ordered elements
- \bullet $S \leftarrow \mathbf{Set}$ of all nodes with no incoming edge

Assignment Project Exam Help

https://powcoder.com

 \bullet remove edge e from the graph;

Add WeChat powcoder

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

Now it should be easy to use the Greedy Strategy to solve the problem of finding the fewest number of towers which you must equip with seismic sensors, so that all emergency transmission towers get activated.

- $L \leftarrow \text{Empty list that will contain ordered elements}$
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help • remove a node u from S:

- $\begin{array}{c} \mathbf{\hat{h}} \mathbf{d} \mathbf{d} \ u \ \text{to tail of } L; \\ \mathbf{\hat{h}} \mathbf{tps:/powcoder.com} \\ \mathbf{\hat{h}} \mathbf{d} \mathbf{n} \ \text{ode } v \ \mathbf{\hat{h}} \mathbf{h} \ \text{an edge } e = (u,v) \ \text{from } u \ \text{to } v \ \mathbf{do} \end{array}$
 - remove edge e from the graph;

Add WeChat powcoder

• if graph has edges left, then return error (graph has at least one cycle)

- \bullet $L \leftarrow$ Empty list that will contain ordered elements
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help

- $\begin{array}{c} \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} & \mathbf{n} \\ \mathbf{n} \\ \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} \\ \mathbf{n} & \mathbf{n} \\ \mathbf{n} \\ \mathbf{n} & \mathbf{n} \\ \mathbf{n} \\ \mathbf{n} \\ \mathbf{n} & \mathbf{n} \\ \mathbf{$
 - remove edge *e* from the graph;

Add WeChat powcoder

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

Now it should be easy to use the Greedy Strategy to solve the problem of finding the fewest number of towers which you must equip with seismic sensors, so that all emergency transmission towers get activated.

- $L \leftarrow$ Empty list that will contain ordered elements
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help • remove a node u from S:

- $\begin{array}{c} \mathbf{add} \ u \ \text{to tail of } L; \\ \mathbf{https:/powcoder.com} \\ \mathbf{for} \ \text{each node} \ v \ \mathbf{with} \ \text{an edge} \ e = (u,v) \ \text{from} \ u \ \text{to} \ v \ \mathbf{do} \end{array}$
- - remove edge e from the graph;

Addha Weendaning the province rounder

• if graph has edges left, then return error (graph has at least one cycle)

- $L \leftarrow$ Empty list that will contain ordered elements
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help • remove a node u from S:

 $\begin{array}{c} \mathbf{add} \ u \ \text{to tail of } L; \\ \mathbf{https:/powcoder.com} \\ \mathbf{for} \ \text{each node} \ v \ \mathbf{with} \ \text{an edge} \ e = (u,v) \ \text{from} \ u \ \text{to} \ v \ \mathbf{do} \end{array}$

- - remove edge e from the graph;

Add We Chia deptower order

• if graph has edges left, then return error (graph has at least one cycle)

- $L \leftarrow \text{Empty list that will contain ordered elements}$
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help • remove a node u from S:

- $\begin{array}{c} \mathbf{add} \ u \ \text{to tail of } L; \\ \mathbf{https:/powcoder.com} \\ \mathbf{for} \ \text{each node} \ v \ \mathbf{with} \ \text{an edge} \ e = (u,v) \ \text{from} \ u \ \text{to} \ v \ \mathbf{do} \end{array}$
 - remove edge e from the graph;

Add has Weet inclaim a telegration were conder

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

- $L \leftarrow$ Empty list that will contain ordered elements
- $S \leftarrow \text{Set of all nodes with no incoming edge}$

Assignment Project Exam Help

- $\begin{array}{c} \mathbf{add} \ u \ \text{to tail of } L; \\ \mathbf{https:/powcoder.com} \\ \mathbf{for} \ \text{each node} \ v \ \mathbf{with} \ \text{an edge} \ e = (u,v) \ \text{from} \ u \ \text{to} \ v \ \mathbf{do} \end{array}$
 - remove edge e from the graph;

Add half be included a second contract of the contract of the

• if graph has edges left, then return error (graph has at least one cycle)

else return L (a topologically sorted order)

Now it should be easy to use the Greedy Strategy to solve the problem of finding the fewest number of towers which you must equip with seismic sensors, so that all emergency transmission towers get activated.

An augmented Priority Queue

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

- We will store in heaps vertices of graphs with keys computed in various ways hit properties we will label them with positive integers 1 to nhttps://powcoder.com
- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besi Adday We Chatth powcoder array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

An augmented Priority Queue

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

Assignments presenter of a cthe et xiami Help A[2i] and the right child in A[2i+1].

- We will store in heaps vertices of graphs with keys computed in various ways hitches://powcoder.com
- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besi Adday We Chatth powcoder array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

Assignments eterojes the Exiami Help

- We will store in heaps vertices of graphs with keys computed in various ways if a graph has η vertices we will label them with positive integers 1 to n
- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besi Adday We Chatth powcoder array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

Assignments etclic the Examily Holp

- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besi Adday We Chatth powcoder array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

Assignments eterojes the Exiami Help

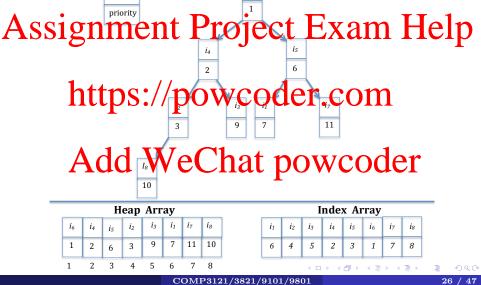
- We will store in heaps vertices of graphs with keys computed in various ways if a graph has n vertices we will label them with positive integers 1 to nhttps://powcoder.com
- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besides the tray Whenepel at the Dap, We will be array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

• We will need a priority queue which allows an efficient change of the key of an element in the queue, so we first need to extend the Heap data structure.

Assignments etclic the Examily Holp

- Thus, every element of A is of the form (i, k(i)) where k(i) is the key of element i.
- Besides the tray Whenepe at the Dap, We wife array P of the same length which stores the position of elements in the heap; thus A[P[i]] = (i, k(i)).
- Changing the key of an element i is now an $O(\log n)$ operation: we look up its position P[i] in A, change the key of the element in A[P[i]] and then perform the Heappify operation to make sure the Heap property is being preserved.

Item



Assignment Project Exam Help

• Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.

• we https://powcoder.com

• For simplicity, we will assume that for every $u \in V$ there is a path from v to u.

Add WeChat powcoder

• This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

Some of the most important applications of the Greedy Strategy are in Assignment Project Exam Help

• Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.

• we https://powcoder.com

• For simplicity, we will assume that for every $u \in V$ there is a path from v to u.

Add WeChat powcoder

 This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

Some of the most important applications of the Greedy Strategy are in Assignment Project Exam Help

- Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.
- we https://powcoder.com
- For simplicity, we will assume that for every $u \in V$ there is a path from v to u.
- Add WeChat powcoder
- This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

- Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.
- we https://powcoder.com
- For simplicity, we will assume that for every $u \in V$ there is a path from v to u.
- Add WeChat powcoder
- This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

- Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.
- we https://powcoder.com
- For simplicity, we will assume that for every $u \in V$ there is a path from v to u.
- The Ask ded find for each tate prosyncheder.
- This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

- Assume we are given a directed graph G = (V, E) with **non-negative** weight $w(e) \ge 0$ assigned to each edge $e \in E$.
- we https://powcoder.com
- For simplicity, we will assume that for every $u \in V$ there is a path from v to u.
- The Add of the Chate power order.
- This is accomplished by a very elegant greedy algorithm by Edsger Dijkstra already in 1959!

- We first prove a simple fact about shortest paths.
- Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

https://powcoder.com

W

- We Add We Chat powcoder from v
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z
- Contradiction!

4 D F 4 B F 4 B F

- We first prove a simple fact about shortest paths.
- Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

https://powcoder.com

W

- We Add We Chat powcoder from v
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z
- Contradiction!

- We first prove a simple fact about shortest paths.
- Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

https://powcoder.com

W

- We claim that for every criteria on that path the shortes path from v to w is just the truncation of the shortest path from v to z, enting at w.
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z
- Contradiction

- We first prove a simple fact about shortest paths.
- Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

https://powcoder.com

W

- We claim that for every artex σ on that path the shortes path from v to w is just the truncation of the shortest path from v to z, eneing at w.
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z
- Contradiction!

- We first prove a simple fact about shortest paths.
- ullet Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

https://powcoder.com

W

- We claim that for every artex σ on that path the shortes path from v to w is just the truncation of the shortes path from v to z; enting at w.
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z.

Contradiction!

- We first prove a simple fact about shortest paths.
- ullet Consider a shortest path p in G from a vertex v to a vertex z (shown in dark blue).

Assignment Project Exam Help

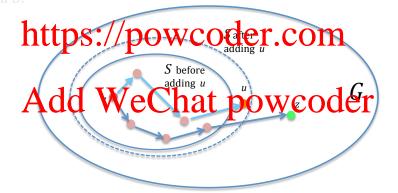
https://powcoder.com

W

- We claim that for every artex σ on that path the shortes path from v to w is just the truncation of the shortes path from v to z; enting at w.
- Assume the opposite, that there is a shorter path from v to w (shown in dashed light blue) which is not an initial segment of the shortest path from v to z.
- However, in that case we could remove the part of the shortest path between v and z which is between v and w and replaced it with the light blue shorter path from v to w, thus obtaining a shorter path from v to z.
- Contradiction!

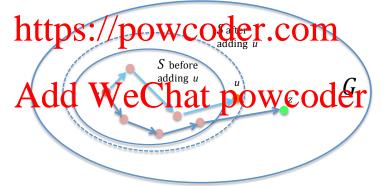
28 / 47

• The algorithm builds a set S of vertices for which the shortest path has been already established, starting with a single source vertex $S=\{v\}$ and adding one vertex at a time.



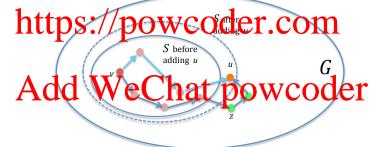
• The algorithm builds a set S of vertices for which the shortest path has been already established, starting with a single source vertex $S=\{v\}$ and adding one vertex at a time.

A Schement Instruction we de the eleveratory Switch has the shortest path from v to u with all intermediate vertices already in S.



- Why does this produce a shortest path from v to u in G?
- Assume the opposite, that there exists a shorter path from v to u in G. ssignment Project Exam Help

 Let Soe the first vertex outside Sas it was just prior to addition of un

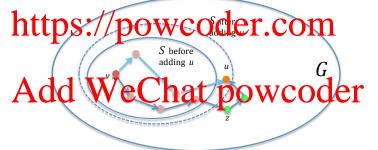


• Since there are no negative weight edges the path from v to such z would

- Why does this produce a shortest path from v to u in G?
- Assume the opposite, that there exists a shorter path from v to u in G.

 Assume the opposite, that there exists a shorter path from v to u in G.

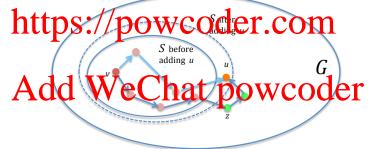
 By our choice of u such a pth cannot be entirely in S Help
 - Let Σ be the first vertex outside S' (as it was just prior to addition of u) on such a shortest path.



• Since there are no negative weight edges the path from v to such z would be shorter than the path from v to u, contradicting our choice of u.

- Why does this produce a shortest path from v to u in G?
- Assume the opposite, that there exists a shorter path from v to u in G.

 By our choice of u such a pth cannot be entrely in S Help
 - Let \mathcal{L} be the first vertex outside \mathcal{L} (as it was just prior to addition of u) on such a shortest path.

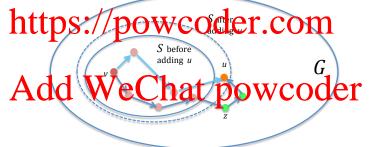


• Since there are no negative weight edges the path from v to such z would be shorter than the path from v to u, contradicting our choice of u.

- Why does this produce a shortest path from v to u in G?
- Assume the opposite, that there exists a shorter path from v to u in G.

 By our choice of u such a pth cannot be entirely in S Help

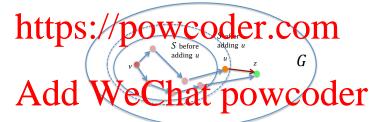
Let \mathcal{E} be the first vertex outside \mathcal{E} (as it was just prior to addition of u) on such a shortest path.



• Since there are no negative weight edges the path from v to such z would be shorter than the path from v to u, contradicting our choice of u.

- How is this construction done efficiently?
- Initially all vertices except v are placed in a heap based priority queue with additional Position array, with the weight w(v,u) if $(v,u) \in E$ or ∞

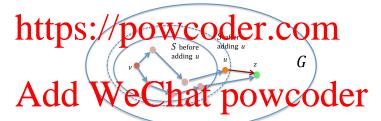
Assignment Project Exam Help vertices on such a path in S.



- We always pop the element u from the priority queue which has the smallest key and add it to set S.
- We then look for all elements $z \in V \setminus S$ for which $(u, z) \in E$ and if $\mathrm{lh}_{S,v}(u) + w(u, z) < \mathrm{lh}_{S,v}(z)$ we update the key of z to the value $\mathrm{lh}_{S,v}(u) + w(u, z)$.

- How is this construction done efficiently?
- Initially all vertices except v are placed in a heap based priority queue with additional Position array, with the weight w(v,u) if $(v,u) \in E$ or ∞

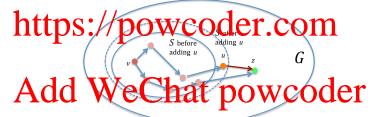
Assignment Project Exam Help vertices on such a path in S.



- We always pop the element u from the priority queue which has the smallest key and add it to set S.
- We then look for all elements $z \in V \setminus S$ for which $(u, z) \in E$ and if $\mathrm{lh}_{S,v}(u) + w(u, z) < \mathrm{lh}_{S,v}(z)$ we update the key of z to the value $\mathrm{lh}_{S,v}(u) + w(u, z)$.

- How is this construction done efficiently?
- Initially all vertices except v are placed in a heap based priority queue with additional Position array, with the weight w(v,u) if $(v,u) \in E$ or ∞ as the key;

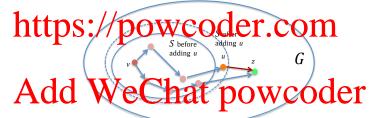
as the key; S_s we progress that they of earth energy will be X_s will be X_s will be X_s with the S_s be S_s which has all intermediate vertices on such a path in S_s .



- We always pop the element u from the priority queue which has the smallest key and add it to set S.
- We then look for all elements $z \in V \setminus S$ for which $(u, z) \in E$ and if $\mathrm{lh}_{S,v}(u) + w(u, z) < \mathrm{lh}_{S,v}(z)$ we update the key of z to the value $\mathrm{lh}_{S,v}(u) + w(u, z)$.

- How is this construction done efficiently?
- Initially all vertices except v are placed in a heap based priority queue with additional Position array, with the weight w(v,u) if $(v,u) \in E$ or ∞ as the key:

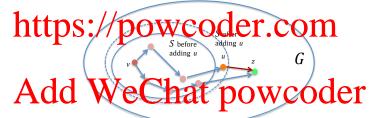
as the key; S_1 we progress that V_2 is a superscript V_3 will be V_4 by V_4 and V_4 of the shortest path from V_4 to V_4 which has all intermediate vertices on such a path in S_4 .



- We always pop the element u from the priority queue which has the smallest key and add it to set S.
- We then look for all elements $z \in V \setminus S$ for which $(u, z) \in E$ and if $\mathrm{lh}_{S,v}(u) + w(u, z) < \mathrm{lh}_{S,v}(z)$ we update the key of z to the value $\mathrm{lh}_{S,v}(u) + w(u, z)$.

- How is this construction done efficiently?
- Initially all vertices except v are placed in a heap based priority queue with additional Position array, with the weight w(v,u) if $(v,u) \in E$ or ∞ as the key:

as the key; S_1 we place that S_2 is the key; S_3 we place that S_2 is the shortest path from to u which has all intermediate vertices on such a path in S.



- We always pop the element u from the priority queue which has the smallest key and add it to set S.
- We then look for all elements $z \in V \setminus S$ for which $(u, z) \in E$ and if $\mathrm{lh}_{S,v}(u) + w(u, z) < \mathrm{lh}_{S,v}(z)$ we update the key of z to the value $\mathrm{lh}_{S,v}(u) + w(u, z)$.

• Why is this enough; i.e., why the only shortest paths which could have changed have u as the last vertex in S?

- Assume opposite that the shortest path to a vertex x has changed when u was added, and that instead of u another node y was the last vertex before x on such a rew shortest path.
- before of such they shortest path.

 How Adds now echat powcoder path to y with a vertex u on that path not belonging to set S before adding u.
- If graph G has n vertices and m edges, then each edge is inspected only once, when it is an outgoing edge from the most recently added vertex; updating a vertex key takes $O(\log n)$ many steps.
- Thus, in total, the algorithm runs in time $O(m \log n)$.

• Why is this enough; i.e., why the only shortest paths which could have changed have u as the last vertex in S?

- Assume opposite that the shortest path to a vertex x has changed when u was added, and that instead of u another node y was the last vertex before Δx of such X be shortest path.
- before x or such New shortest path.
 However, this is not possible begand it will have of startest path to y with a vertex u on that path not belonging to set S before adding u.
- If graph G has n vertices and m edges, then each edge is inspected only once, when it is an outgoing edge from the most recently added vertex; updating a vertex key takes $O(\log n)$ many steps.
- Thus, in total, the algorithm runs in time $O(m \log n)$.

• Why is this enough; i.e., why the only shortest paths which could have changed have u as the last vertex in S?

- Assume opposite that the shortest path to a vertex x has changed when u was added, and that instead of u another node y was the last vertex before x on such a rew shortest path.
- before x or such Thew shortest path.
 However this is not possible behalf it will Mour Offertest path to y with a vertex u on that path not belonging to set S before adding u.
- If graph G has n vertices and m edges, then each edge is inspected only once, when it is an outgoing edge from the most recently added vertex; updating a vertex key takes $O(\log n)$ many steps.
- Thus, in total, the algorithm runs in time $O(m \log n)$.

• Why is this enough; i.e., why the only shortest paths which could have changed have u as the last vertex in S?

- Assume opposite that the shortest path to a vertex x has changed when u was added, and that instead of u another node y was the last vertex before x of such TeV shortest path.
- However, this is not possible because it will Mode Official path to y with a vertex u on that path not beloaging to set S before adding u.
- If graph G has n vertices and m edges, then each edge is inspected only once, when it is an outgoing edge from the most recently added vertex; updating a vertex key takes $O(\log n)$ many steps.
- Thus, in total, the algorithm runs in time $O(m \log n)$.

• Definition: A minimum spanning tree T of a connected graph G is a subgraph of G (with the same set of vertices) which is a tree, and among all such trees it is of minipal total length of Exam Help

• Lemma: Let G be a connected graph with all lengths of edges E of G distinct and S a non empty proper subset of the set of all vertices V of G. Assume that e = (u, v) is an edge such that $u \in S$ and $v \notin S$ and is of minihttps://powcoder.com

• Proof:



4□▶ 4□▶ 4□▶ 4□▶ □ 900

• **Definition:** A minimum spanning tree T of a connected graph G is a subgraph of G (with the same set of vertices) which is a tree, and among all such trees it is of minimal total length of all edges in T Help

• Lemma: Let G be a connected graph with all lengths of edges E of G distinct and S a non empty proper subset of the set of all vertices V of G. Assume that e = (u, v) is an edge such that $u \in S$ and $v \notin S$ and is of minimal length among all the edges having this property. Then e must belong the edge manning all the edges having this property.

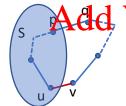
Proof:



• **Definition:** A minimum spanning tree T of a connected graph G is a subgraph of G (with the same set of vertices) which is a tree, and among all such trees it is of minimal total length of a redges in T Help

• Lemma: Let G be a connected graph with all lengths of edges E of G distinct and S a non empty proper subset of the set of all vertices V of G. Assume that e = (u, v) is an edge such that $u \in S$ and $v \notin S$ and is of minimal length among all the edges having this property. Then e must belong the edge manning that G is G.

• Proof:



winimum spanning tree which does not contain such an edge e = (u, v).

• Since T is a spanning tree, there exists a path from u to v within such a tree.

• **Definition:** A minimum spanning tree T of a connected graph G is a subgraph of G (with the same set of vertices) which is a tree, and among all such trees it is of minimal total length of all edges in T Help

• Lemma: Let G be a connected graph with all lengths of edges E of G distinct and S a non empty proper subset of the set of all vertices V of G. Assume that e = (u, v) is an edge such that $u \in S$ and $v \notin S$ and is of minimal length among all the edges having this property. Then e must belong the edge manning all the edges having this property.

• Proof:



• Since T is a spanning tree, there exists a path from u to v within such a tree.

Assignment Project Exam Help

https://powcoder.com

removing the edge (p,q).

• However, the edge (p,q) belongs

• Since A \in add W, the Chat powcoder is certain point.

• Assume that p is the last vertex on this path which is in S and $q \notin S$ the vertex immediately after p on that path.

would result in a spanning tree of smaller weight, contradicting our assumption that T is a minimum spanning tree.

Assignment Project Exam Help S. However, the edge (p,q) belongs to T and must have a length edge (p,q) (Help) Let T and must have a length edge (p,q) (Help) S.

https://powcoder.com

removing the edge (p, q)

- Since A & Aid W. the Chat powcoder pathems are left at the certain point.
- Assume that p is the last vertex on this path which is in S and $q \notin S$ the vertex immediately after p on that path.

4 D > 4 D > 4 E > 4 E > E 990

Assignment Prostremining Large (p,q) belongs to T and must have a length larger than the edge (u,v) (which can be end in S and one end outside S.

Adding the edge e = (u,v) to https://powcoder.com

• Since A & Add W. & Chat powcoder sertain point.

• Assume that p is the last vertex on this path which is in S and $q \notin S$ the vertex immediately after p on that path.

would result in a spanning tree of smaller weight, contradicting our assumption that T is a minimum spanning tree.

removing the edge (p,q).

Assignment Prostremininal Again to S and one end outside S.

Adding the edge S and one end outside S.

Adding the edge S and S are altastic from S and S and S are altastic from S are altastic from S and S are altastic from S are altastic from S and S are altastic from S are altastic f

• Since A \in add \in A \in Char powcoder |
certain point.

• Assume that p is the last vertex on this path which is in S and $q \notin S$ the vertex immediately after p on that path.

would result in a spanning tree of smaller weight, contradicting our assumption that T is a minimum spanning tree.

removing the edge (p, q).

• However, the edge (p,q) belongs to T and must have a length ignment Project than he edge (y, y) (which the continual larger than he edge (• Adding the edge e = (u, v) to ttps://powsicla.gath.from.u.to.u.produces

- certain point.
- Assume that p is the last vertex on this path which is in S and $q \notin S$ the vertex immediately after p on that path.

However, since by assumption the would result in a spanning tree of smaller weight, contradicting our assumption that T is a minimum spanning tree.

removing the edge (p, q).

Assignment Project Exam Help We order the edges E in a non-decreasing order with respect to their

• We https://powcoder.com

• An edge e_i is added at a round i of construction whenever its addition

• If a Add a We Chat powcoder

- weights.
- We https://powcoder.com
- An edge e_i is added at a round i of construction whenever its addition
- If a Add a We Chat powcoder

- weights.
- We hittps by adpowe oder reconnection.
- An edge e_i is added at a round i of construction whenever its addition
- If a Add a We Chat powcoder

- weights.
- We hittps://poweoder.comuction.
- An edge e_i is added at a round i of construction whenever its addition does not introduce a cycle in the graph constructed thus far.
- If a Add a We Chat powcoder

- weights.
- We hittps by adpowe oder reconnection.
- An edge e_i is added at a round i of construction whenever its addition does not introduce a cycle in the graph constructed thus far.
- If adding a lege was the dute at the Da Wels the test

- weights.
- We hittps by adpowe oder reconnection.
- An edge e_i is added at a round i of construction whenever its addition does not introduce a cycle in the graph constructed thus far.
- If adding a lege was the dute at the Da Wels the test
- The process terminates when the list of all edges has been exhausted.

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

Assignment Project Exam Help

- Consider the set S of all vertices w such that there exists a path from uto unising only the yelset of edges that have been added by the Kruskal algohttps://powcoder.comed.
- \bullet Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not Add WeChat powcoder
- Thus, the set of edges produced by the Kruskal algorithm is a subset of
- But the structure produced by the Kruskal algorithm is clearly cycle-free

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

Arssigninent Project Exam Help

- Consider the set S of all vertices w such that there exists a path from uto unising only the subset of edges that have been added by the Kruskal algorithms. Powcoder.comed.
- \bullet Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not Add WeChat powcoder
- Thus, the set of edges produced by the Kruskal algorithm is a subset of
- But the structure produced by the Kruskal algorithm is clearly cycle-free

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from uto unising only the subset of edges that have been added by the Kruskal algorithms. Powcoder.comed.
- \bullet Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not Add WeChat powcoder
- Thus, the set of edges produced by the Kruskal algorithm is a subset of
- But the structure produced by the Kruskal algorithm is clearly cycle-free

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to wrusing only the subset of edges that have been added by the Kruskal algorith hour Sust be of the die of the control of
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not Add WeChat powcoder
- Thus, the set of edges produced by the Kruskal algorithm is a subset of
- But the structure produced by the Kruskal algorithm is clearly cycle-free

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to wrusing only the subset of edges that have been added by the Kruskal algorithm with Sust Melor the dige () (A) has been alded.
- Then clearly $u \in S$ but $v \notin S$.
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not Add WeChat powcoder
- Thus, the set of edges produced by the Kruskal algorithm is a subset of
- But the structure produced by the Kruskal algorithm is clearly cycle-free

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to w using only the subset of edges that have been added by the Kruskal algorithm units ust be of three (u, v) has been added.
- Then clearly $u \in S$ but $v \notin S$.
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not forming advels.
- Consequently, edge e = (u, v) is the shortest one with such a property and by the previous lemma it must belong to every spanning tree.
- Thus, the set of edges produced by the Kruskal algorithm is a subset of the set of edges of every spanning tree.
- But the structure produced by the Kruskal algorithm is clearly cycle-free and it spans the graph, otherwise another edge could be added.

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to w using only the subset of edges that have been added by the Kruskal algorithm unit Susy be of the vertice of the land of
- Then clearly $u \in S$ but $v \notin S$.
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not forming acycle.
- Consequently, edge e = (u, v) is the shortest one with such a property and by the previous lemma it must belong to every spanning tree.
- Thus, the set of edges produced by the Kruskal algorithm is a subset of the set of edges of every spanning tree.
- But the structure produced by the Kruskal algorithm is clearly cycle-free and it spans the graph, otherwise another edge could be added.

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to w using only the subset of edges that have been added by the Kruskal algorithm with Sust below the whole (u, w) have a fided.
- Then clearly $u \in S$ but $v \notin S$.
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not forming advels.
- Consequently, edge e = (u, v) is the shortest one with such a property and by the previous lemma it must belong to every spanning tree.
- Thus, the set of edges produced by the Kruskal algorithm is a subset of the set of edges of every spanning tree.
- But the structure produced by the Kruskal algorithm is clearly cycle-free and it spans the graph, otherwise another edge could be added.

Claim: The Kruskal algorithm produces a minimal spanning tree, and if all weights are distinct, then such a Minimum Spanning Tree is unique.

- Consider the set S of all vertices w such that there exists a path from u to w using only the subset of edges that have been added by the Kruskal algorithm unit Susy be of the vertice of the land of
- Then clearly $u \in S$ but $v \notin S$.
- Until this moment no edge with one end in S and the other outside S has been considered because otherwise it would have been added, not forming acycle.
- Consequently, edge e = (u, v) is the shortest one with such a property and by the previous lemma it must belong to every spanning tree.
- Thus, the set of edges produced by the Kruskal algorithm is a subset of the set of edges of every spanning tree.
- But the structure produced by the Kruskal algorithm is clearly cycle-free and it spans the graph, otherwise another edge could be added.

To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets pelements, called the Union-Find. Help

① MakeUnionFind(S), which, given a set S returns a structre in which all elements are placed into distinct singleton sets. Such an

https://powcoder.com

② FIND(v), which, given an element v returns the (label of the) set to which v belongs. Such operation should run either in time O(1) or

Add We Chat powcoder
UNION(A, B), which, given (the labels of) two sets A, B changes the

To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

ASSINGTANTIC IN has to kup de Get or erakaam Help

① MakeunionFind(S), which, given a set S returns a structre in which all elements are placed into distinct singleton sets. Such an

https://powcoder.com

FIND(v), which, given an element v returns the (label of the) set to which v belongs. Such operation should run either in time O(1) or

Add We Chat powcoder
UNION(A, B), which, given (the labels of) two sets A, B changes the

To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Schantic In tast of the Color of the Union-Find.

Help

• MakeUnionFind(S), which, given a set S returns a structure in which all elements are placed into distinct singleton sets. Such an operation should run in time O(n) where n = |S|.

② FIND(v), which, given an element v returns the (label of the) set to which v belongs. Such operation should run either in time O(1) or

Add We Chat powcoder
UNION(A, B), which, given (the labels of) two sets A, B changes the

• To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Help

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements.

Help

Help

Help

Help

**To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements.

Help

Help

Help

Help

Help

**To efficiently implement the Kruskal algorithm.

Help

**H

• MakeUnionFind(S), which, given a set S returns a structure in which all elements are placed into distinct singleton sets. Such an operation should run in time O(n) where n = |S|.

https://powcoder.com

② FIND(v), which, given an element v returns the (label of the) set to which v belongs. Such operation should run either in time O(1) or time $O(\log n)$ as we explain later.

time $G(\log n)$ we explain later. Powcoder UNION(A,B), which, given (the labels of) two sets A,B changes the

To efficiently implement the Kruskal algorithm, we need a useful data structure for storing sets of elements, called the Union-Find.

Schannican basto top decet operation Help

• MakeUnionFind(S), which, given a set S returns a structure in which all elements are placed into distinct singleton sets. Such an operation should run in time O(n) where n = |S|.

ntips://powcoder.com

2 FIND(v), which, given an element v returns the (label of the) set to which v belongs. Such operation should run either in time O(1) or time $O(\log n)$ as we explain later.

Add We Chat powcoder

UNION(A, B), which, given (the labels of) two sets A, B changes the

• Note that we do not give the run time of a single Union operation but of a sequence of k consecutive such operations.

Assignment Project Exam Help operations, in this case log k.

- We https://powcoder.com/s graphs, we can assume that the underlying set S is of the form $S = \{1, 2, ..., n\}$.
- of: Add WeChat powcoder
 - ① an array A such that A[i] = j means that i belongs to set labeled j;
 - ② an array B such that B[i] contains the number of elements in the set labeled by i (which can be 0) and pointers to the first and last element of a linked list of elements of the set labeled by i.

• Note that we do not give the run time of a single Union operation but of a sequence of k consecutive such operations.

Assignment and virial leaf of an operation in a sequence of operations, in this case $\log k$.

• We https://powcoder.com/s graphs, we can assume that the underlying set S is of the form $S = \{1, 2, ..., n\}$.

- of: Add WeChat powcoder
 - ① an array A such that A[i] = j means that i belongs to set labeled j;
 - ② an array B such that B[i] contains the number of elements in the set labeled by i (which can be 0) and pointers to the first and last element of a linked list of elements of the set labeled by i.

• Note that we do not give the run time of a single Union operation but of a sequence of k consecutive such operations.

Assing more it that virtue is considered to the constant of t

• We vill tabel each set by one of its elements. Since we will use the Union Find data structure to keep that itsels of vertiles of graphs, we can assume that the underlying set S is of the form $S = \{1, 2, \ldots, n\}$.

of: Add WeChat powcoder

- ① an array A such that A[i] = j means that i belongs to set labeled j;
- ② an array B such that B[i] contains the number of elements in the set labeled by i (which can be 0) and pointers to the first and last element of a linked list of elements of the set labeled by i.

• Note that we do not give the run time of a single Union operation but of a sequence of k consecutive such operations.

Assignment in a virtie of an operation in a sequence of operations, in this case $\log k$.

- We will debe each set by one of its elements. Since we will use the Union Find data structure of Kept tack at the Source of graphs, we can assume that the underlying set S is of the form $S = \{1, 2, \ldots, n\}$.
- The simplest implementation of the Union-Find data structure consists of: Add we chat powcoder
 - an array A such that A[i] = j means that i belongs to set labeled j;
 - 2 an array B such that B[i] contains the number of elements in the set labeled by i (which can be 0) and pointers to the first and last element of a linked list of elements of the set labeled by i.

• Note that we do not give the run time of a single Union operation but of a sequence of k consecutive such operations.

Assignment and virtual content of an operation in a sequence of operations, in this case $\log k$.

- We will debe each set by one of its elements. Since we will use the Union Find data structure of Kept tack at the Source of graphs, we can assume that the underlying set S is of the form $S = \{1, 2, \ldots, n\}$.
- The simplest implementation of the Union-Find data structure consists of: Add WeChat powcoder
 - an array A such that A[i] = j means that i belongs to set labeled j;
 - ② an array B such that B[i] contains the number of elements in the set labeled by i (which can be 0) and pointers to the first and last element of a linked list of elements of the set labeled by i.

• Union(i,i) of two sets labeled by i and j, respectively, is defined as

ssignment Project Exam Help number of elements in the set labeled by j then labels in array A of all elements in the set labeled by j is changed to i and array B is

https://powcoder.com

• if the set labeled by i has more elements than the set labeled by i,

Add WeChat powcoder

• Note that this definition implies that if UNION(i, j) changes the label of

• Union(i,i) of two sets labeled by i and j, respectively, is defined as

ssignment Project Exam Help number of elements in the set labeled by j then labels in array A of all elements in the set labeled by j is changed to i and array B is

https://powcoder.com

• if the set labeled by i has more elements than the set labeled by i, then the labels of all elements in the set labeled by i are changed to Add Wechacordingly. Dowcoder

• Note that this definition implies that if UNION(i, j) changes the label of

• Union(i,j) of two sets labeled by i and j, respectively, is defined as

ssignment Project Exam Help number of elements in the set labeled by j then labels in array A of all elements in the set labeled by j is changed to i and array B is

https://powcoder.com

• if the set labeled by i has more elements than the set labeled by i, then the labels of all elements in the set labeled by i are changed to

Add Wechat powcoder

• Note that this definition implies that if UNION(i, j) changes the label of the set containing an element m, then the new set containing m will have at least twice the number of elements of the set which contained m before the UNION(i, j) operation.

Any sequence of k initial consecutive UNION operations can touch at As a light diagram of S(w) the happens if all thou operation were p

• Thus, the set containing an element m after k initial consecutive UNION must have at most 2k elements.

https://powcoder.com

containing m at least doubles the size of the set containing that element the label of the set containing m could change fewer than $\log 2k$ many times k

Add WeChat powcoder

Thus, since we have at most 2k element pany sequence of k initial

Thus, since we have at most 2k elements, any sequence of k initial consecutive UNION operations will have in total fewer than $2k \log 2k$ many label changes in A and each UNION operation changes just a few pointers in B and adds up the sizes of sets.

Any sequence of k initial consecutive UNION operations can touch at As a light diagram of S(w) the happens if all thou operation were p

• Thus, the set containing an element m after k initial consecutive UNION must have at most 2k, elements.

https://powcoder.com

containing m at least doubles the size of the set containing that element the label of the set containing m could change fewer than $\log 2k$ many times.

Add WeChat powcoder

Thus, since we have at most 2k element pany sequence of k initial

Thus, since we have at most 2k elements, any sequence of k initial consecutive UNION operations will have in total fewer than $2k \log 2k$ many label changes in A and each UNION operation changes just a few pointers in B and adds up the sizes of sets.

Any sequence of k initial consecutive UNION operations can touch at As most 2k elements of S(w) happens if all thion operation. Were probabilistically project EX and Help

• Thus, the set containing an element m after k initial consecutive UNION must have at most 2k elements.

Since every Union operation which changes the label of the set containing m at least doubles the size of the set containing that element, the label of the set containing m could change fewer than $\log 2k$ many times.

Add WeChat powcoder

Thus, since we have at most 2k element pany sequence of k initial

• Thus, since we have at most 2k elements, any sequence of k initial consecutive UNION operations will have in total fewer than $2k \log 2k$ many label changes in A and each UNION operation changes just a few pointers in B and adds up the sizes of sets.

Any sequence of k initial consecutive UNION operations can touch at As a light diagram of $S_k(w)$ by happens if all things operation were light diagram of $S_k(w)$ by happens if all things operation were light diagram of $S_k(w)$ by happens if all things of $S_k(w)$ by happens if all things of $S_k(w)$ by happens if all things of $S_k(w)$ by the sequence of $S_$

• Thus, the set containing an element m after k initial consecutive UNION must have at most 2k elements.

• Since every Union operation which changes the label of the set containing m at least doubles the size of the set containing that element, the label of the set containing m could change fewer than $\log 2k$ many times.

Add WeChat powcoder

Thus, since we have at most 2k element, any sequence of k initial

• Thus, since we have at most 2k elements, any sequence of k initial consecutive Union operations will have in total fewer than $2k \log 2k$ many label changes in A and each Union operation changes just a few pointers in B and adds up the sizes of sets.

Assignment Project Exam Help complexity of $O(k \log k)$.

- https://powcoder.com
- See the textbook for an Union-Find data structure based on pointers and path compression, which further reduces the amortised complexity of the UNIOAnddoweChatsippowcoderFIND operAddoweChatsippowcoder

Assignment Project Exam Help complexity of $O(k \log k)$.

- Such Union-Find data structure is good enough to get the sharpest possible being on the purpose the fraction alter than alter than alter than 1
- See the textbook for an Union-Find data structure based on pointers and path compression, which further reduces the amortised complexity of the UNIOA of the Chatsipto work of the UNIOA of the Chatsipto work of the Chat

Assignment Project Exam Help complexity of $O(k \log k)$.

- Such Union-Find data structure is good mough to get the sharpest possible by the production of all or thin.
- See the textbook for an Union-Find data structure based on pointers and path compression, which further reduces the amortised complexity of the UNION operation with the cost of increasing the complexity of the FIND operation from O(V) to O(

We now use the previously described Union-Find data structure to efficiently implement the Kruskal algorithm on a graph G = (Y, F) with ASSISIAMENT. Project Exam Help

• We first have to sort m edges of graph G which takes time $O(m \log m)$. Since $m \le n^2$, this step of the algorithm also takes

https://powcoder.com

- As we progress through the Kruskal algorithm execution, we will be making connected components which will be merged into larger connected components until all vertices belong to a single connected compared two chat powcoder keep track that is a single connected to the contract of the
- For each edge e = (v, u) on the sorted list of edges we use two FIND operations, FIND(u) and FIND(v) to determine if vertices u and v belong to the same component.

• We now use the previously described Union-Find data structure to efficiently implement the Kruskal algorithm on a graph G = (Y, F) with ASSISIANCIAL Project Exam Help

- We first have to sort m edges of graph G which takes time $O(m \log m)$. Since $m \le n^2$, this step of the algorithm also takes $O(m \log n^2) = S(m \log n) OWCOder.COM$
- As we progress through the Kruskal algorithm execution, we will be making connected components which will be merged into larger connected components until all vertices belong to a single connected compared tweethat powcoder keep track that is a single connected to the contract of the connected compared to the connected connected to the connected to the connected connected to the connected connected to the connected to t
- For each edge e=(v,u) on the sorted list of edges we use two Find operations, $\operatorname{Find}(u)$ and $\operatorname{Find}(v)$ to determine if vertices u and v belong to the same component.

We now use the previously described Union-Find data structure to efficiently implement the Fuskal algorithm on a graph G = (Y, F) with Assessment Froject Exam Help

- We first have to sort m edges of graph G which takes time $O(m \log m)$. Since $m \le n^2$, this step of the algorithm also takes $O(m \log n^2) = S(m \log n) OWCOder.COM$
- As we progress through the Kruskal algorithm execution, we will be making connected components which will be merged into larger connected components until all vertices belong to a single connected component for this purpose verse Union First data structure to keep track the connected components constructed thus far.
- For each edge e = (v, u) on the sorted list of edges we use two FIND operations, FIND(u) and FIND(v) to determine if vertices u and v belong to the same component.

• We now use the previously described Union-Find data structure to efficiently implement the Fuskal algorithm on a graph G = (Y, F) with Assessment Froject Exam Help

- We first have to sort m edges of graph G which takes time $O(m \log m)$. Since $m \le n^2$, this step of the algorithm also takes $O(m \log n^2) = S(m \log n) OWCOder.COM$
- As we progress through the Kruskal algorithm execution, we will be making connected components which will be merged into larger connected components until all vertices belong to a single connected component for this purpose verse Union First data structure to keep track the connected components constructed thus far.
- For each edge e=(v,u) on the sorted list of edges we use two FIND operations, FIND(u) and FIND(v) to determine if vertices u and v belong to the same component.

- As suggested the property of the property of the property of the property of the spanning tree being pconstructed and perform UNION(i, j) to place u and v into the same connected component.
 - In thttps://powcoder.com in total

 - We also perform n-1 UNION operations which in total cost $O(n \log n)$.

 Add WeChat powcoder $O(n \log n)$ operations; thus, we obtain an overall time complexity of $O(m \log n)$.

- As sthe property to the range potent itexistippu) = dconstructed and perform UNION(i, j) to place u and v into the same connected component.
 - In the print proved the coting of the coti

 - We also perform n-1 UNION operations which in total cost $O(n \log n)$.

 Add WeChat powcoder noperations; thus, we obtain an overall time complexity of $O(m \log n)$.

- **SETHOLOUP** to the lame temperature identification f(x,y) = f(x,y) to the spanning tree being constructed and perform UNION(i, j) to place u and v into the same connected component.
 - In total to pism / poweroclercottem, in total

 - We also perform n-1 Union operations which in total cost $O(n \log n)$.
 Initial Add WeChat powcoder noperations; thus, we obtain an overall time complexity of $O(m \log n)$.

- Sthotopen to the same parent identification u(u) = 0. Finite u(u) = 0, u(u) = 0, we add edge u(u) to the spanning tree being constructed and perform UNION(i, j) to place u and v into the same connected component.
 - In told the piscip of the property of the pr
 - We also perform n-1 UNION operations which in total cost $O(n \log n)$.
 - Initial sorting of edges takes of mag m PO(m log n) operations; thus, we obtain an overall time complexity of $O(m \log n)$.

k-clustering of maximum spacing

Assignment Project Exam Help

• Task: Partition the vertices of G into k disjoint subsets so that the minimal distance between two points belonging to different sets of the partition is as large as possible. Thus, we want a partition into k disjoint sets which are as far a https://powcoder.com

• Solution: Sort the edges in an increasing order and start performing the usual Kruskal's algorithm for building a minimal spanning tree, but stop when you obtain a connected to propose that the than a single spanning tree.

Add WeChat powcoder

Proof of optimality: Let d be the distance associated with the first edge of

• **Proof of optimality:** Let d be the distance associated with the first edge of the minimal spanning tree which was not added to our k connected components; it is clearly the minimal distance between two vertices belonging to two of our k connected. Clearly, all the edges included in k many connected components produced by our algorithm are of length smaller or equal to d.

k-clustering of maximum spacing

Assignmentative Project Exam Help

- Task: Partition the vertices of G into k disjoint subsets so that the minimal distance between two points belonging to different sets of the partition is as large as possible. Thus, we want a partition into k disjoint sets which are as far a large Sprey/powcoder.com
- Solution: Sort the edges in an increasing order and start performing the usual Kruskal's algorithm for building a minimal spanning tree, but stop when you obtain a connected support at the than a single spanning tree.

Add WeChat powcoder

• Proof of optimality: Let d be the distante associated with the first edge of the minimal spanning tree which was not added to our k connected components; it is clearly the minimal distance between two vertices belonging to two of our k connected. Clearly, all the edges included in k many connected components produced by our algorithm are of length smaller or equal to d.

k-clustering of maximum spacing

• Instance: A complete graph G with weighted edges representing distances
Assignment Project Exam Help

• Task: Partition the vertices of G into k disjoint subsets so that the minimal distance between two points belonging to different sets of the partition is as large as possible. Thus, we want a partition into k disjoint sets which are as far a large as possible. DOWCOGET.COM

• Solution: Sort the edges in an increasing order and start performing the usual Kruskal's algorithm for building a minimal spanning tree, but stop when you obtain k cornected components tather than a single spanning tree.

Add WeChat powcoder

• **Proof of optimality:** Let d be the distance associated with the first edge of the minimal spanning tree which was not added to our k connected components; it is clearly the minimal distance between two vertices belonging to two of our k connected. Clearly, all the edges included in k many connected components produced by our algorithm are of length smaller or equal to d.

k-clustering of maximum spacing

• Instance: A complete graph G with weighted edges representing distances

ssignment Project Exam Help • Task: Partition the vertices of G into k disjoint subsets so that the minimal

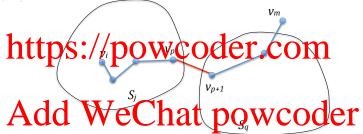
- distance between two points belonging to different sets of the partition is as large as possible. Thus, we want a partition into k disjoint sets which are as far a https://powcoder.com
- Solution: Sort the edges in an increasing order and start performing the usual Kruskal's algorithm for building a minimal spanning tree, but stop when you obtain k connected components tather than a single spanning tree.

• Proof of optimality: Let d be the distance associated with the first edge of the minimal spanning tree which was not added to our k connected components; it is clearly the minimal distance between two vertices belonging to two of our k connected. Clearly, all the edges included in k many connected components produced by our algorithm are of length smaller or equal to d.

k-clustering of maximum spacing

 \bullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

Assignment Project Exam Help



- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_j and $v_{p+1} \notin S_j$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

k-clustering of maximum spacing

 \bullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

As This means that there is a proceed component produced by our regrithment, and respect to the property of th

Add WeChat powcoder

- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_i and $v_{p+1} \notin S_i$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

k-clustering of maximum spacing

 \bullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

As This means that there is a proceed component produced by our regrithment, and respect to the property of th

Add WeChat powcoder

- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_i and $v_{p+1} \notin S_i$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

k-clustering of maximum spacing

 \bullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

This means that there is a proceed component produced by our righting process of the process of

https://powpcoder.com

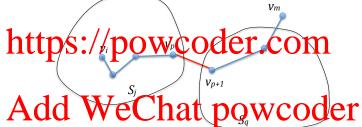
Add WeChat powcoder

- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_j and $v_{p+1} \notin S_j$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

k-clustering of maximum spacing

ullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

This means that there is a connected component produced by our dignithm $v_m \not\in S_1$.



- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_j and $v_{p+1} \notin S_j$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

k-clustering of maximum spacing

 \bullet Consider any partition ${\mathcal S}$ into k subsets different from the one produced by our algorithm.

This means that there is a princeted component produced by our righting $v_m \not\in S$. $v_m \not\in S$

https://powecoder.com Add WeChat powcoder

- Since v_i and v_m belong to the same connected component, there is a path in that component connecting v_i and v_m .
- Let v_p and v_{p+1} be two consecutive vertices on that path such that v_p belongs to S_i and $v_{p+1} \notin S_i$.
- Thus, $v_{p+1} \in S_q$ for some $q \neq j$.

• Note that $d(v_p,v_{p+1}) \leq d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Setwenth material of polyteriol could be considered by a Marit M

• Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.

. w.https://powcoder.com

• We have $O(n^2)$ edges; thus sorting them by weight will take $O(n^2 \log n^2) = O(n^2 \log n)$

Add WeChat powcoder

While running the (partial) Kruskal algorithm we use the UNION-FIND

- While running the (partial) Kruskal algerithm we use the UNION-FIND data structure which requires $O(n^2 \log n)$ steps.
- So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

• Note that $d(v_p, v_{p+1}) \le d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Setwenth material of points reduced by a Harit m

• Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.

. w.https://powcoder.com

• We have $O(n^2)$ edges; thus sorting them by weight will take $O(n^2 \log n^2) = O(n^2 \log n)$

Add We Chat powcoder

While running the (partial) Kruskal alg ithm we use the UNION-FIND

- While running the (partial) Kruskal algaeithm we use the UNION-FIND data structure which requires $O(n^2 \log n)$ steps.
- So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

• Note that $d(v_p,v_{p+1}) \leq d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Setwenth restricted on points reduced by a Harithm e1 p

• Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.

. whattps://powcoder.com

• We have $O(n^2)$ edges; thus sorting them by weight will take $O(n^2 \log n^2) = O(n^2 \log n)$

Add We Chat powcoder

While running the (partial) Kruskal alg ithm we use the UNION-FIND

- While running the (partial) Kruskal algerithm we use the UNION-FIND data structure which requires $O(n^2 \log n)$ steps.
- So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

• Note that $d(v_p, v_{p+1}) \leq d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Sequential parents of the minimal distance d points of the minim

- Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.
- . whattps://powcoder.com
- We have O(n²) edges; thus sorting them by weight will take
 O(n² log n²) O(n² log n²) Chat powcoder
 While running the (partial) Kruskal algorithm we use the UNION-FIND
- While running the (partial) Kruskal algorithm we use the UNION-FINI data structure which requires $O(n^2 \log n)$ steps.
- So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

• Note that $d(v_p, v_{p+1}) \leq d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Sequential parents of the minimal distance d points of the minim

• Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.

. whattps://powcoder.com

• We have $O(n^2)$ edges; thus sorting them by weight will take $O(n^2 \log n^2) = O(n^2 \log n)$ Chat powcoder

• While running the (partial) Kruskal algorithm we use the UNION-FIND data structure which requires $O(n^2 \log n)$ steps.

• So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

• Note that $d(v_p, v_{p+1}) \leq d$ which implies that the distance between these two clusters $S_j, S_q \in \mathcal{S}$ is smaller or equal to the minimal distance d Setworth in parented on points reduced by a light through the produced by a light through the composition of the produced by a light through the composition of the produced by a light through the composition of the produced by a light through the composition of the composition of the produced by a light through the composition of the composition

• Thus, such a partition cannot be a more optimal clustering than the one produced by our algorithm.

. whattps://powcoder.com

• We have $O(n^2)$ edges; thus sorting them by weight will take $O(n^2 \log n^2)$ and $O(n^2 \log n)$ Chat powcoder

- While running the (partial) Kruskal algorithm we use the UNION-FIND data structure which requires $O(n^2 \log n)$ steps.
- So the grand total for the whole algorithm is $O(n^2 \log n)$ many steps.

PUZZLE!!



The Elbonian postal service mandates that boxes to be sent, if not empty, Assignment Protect. Lexand render with the sender. You can send padlocks only if they are locked. How can Bob safely send his teddybear to Alice?

can be used to lock the boxes. However, there is a problem.

OAM.CO.Co.ThoCO.Th.cs are locked (via a padlock) is important as Bob is visiting Elbonia and well as that both Bob and Alice have wishes to send his teddybear to padlocks and boxes. They can also Alice who is staying at wifferen communicate over the phone to agree hotel. Both Boll and Alive take on the trates I Neve are though slible boxes like the one shown on the solutions; one can be called the "AND" picture as well as padlocks which solution, the other can be called the "OR" solution. The "AND" solution requires 4 mail one way services while the "OR" solution requires only 2.