Spaces

Notifications

Search Quora

Add Question

Divide and Rule Dynamic Programming (DP) Algorithms Computer Programming

What is divide and conquer optimization in dynamic programming?

 ${\cal S} \ \text{http://codeforces.com/blog/entry/8219}$

This question previously had details. They are now in a comment.





Ad by DigitalOcean

Starting a new project? Get started for free.

Scale your deployments with a flexible and predictable pricing model. Free for 60 days.

Learn more at try.digitalocean.com

000

2 Answers



Michael Levin, I teach Algorithms: https://goo.gl/MROslP and https://goo.gl/F1vtbF



Updated Apr 5

This is an optimization for computing the values of Dynamic Programming (DP) of the form $dp[i][j] = \min_{k < j} (dp[i-1][k] + C[k+1][j])$ for some arbitrary cost function C[i][j] such that the following property can be proved about this dynamic programming with this cost function. Let's denote by A[i][j] the optimal k for which dp[i][j] = dp[i-1][k] + C[k][j]. The property is that for any i and j, $A[i][j] \leq A[i][j+1]$, that is, the optimal k is monotone on j for fixed i.

An example of such DP is the following problem: given n objects with weights w_1, w_2, \ldots, w_n , divide them into m groups of **consecutive** objects, such that the sum of squares of total weights of the groups is minimal (total weight of the group is the sum of weights of the objects in the group). It can be proved that the optimal k in this problem is monotone on j. The same can be proved if we take cost function $W \log W$ for the total weight W of the group given that all the weights are positive (minimize the sum of $W \log W$ among all distributions of n objects into m groups) or any other **convex function** of the total weight of the group. In these cases, C[i][j] in the DP formulation would be the cost function for a group with objects from i to j inclusive.

The straighforward solution of this DP is $O(mn^2)$, because we need a loop over i (m iterations), a loop over j (n iterations) and a loop over k for each j (n/2 iterations on average). However, given the monotonicity condition $A[i][j] \leq A[i][j+1]$, this DP can be solved in $O(mn\log n)$. More specifically, for each fixed i we will solve the iteration in $O(n\log n)$ instead of $O(n^2)$.

This can be done by the following pseudocode which for fixed i applies divide and conquer on j and keeps range (jleft, jright) of values of j for which we are seeking the answer and the corresponding range (kleft, kright) of the possible values for optimal k when j is in the range (jleft, jright):

```
1 def ComputeDP(i, jleft, jright, kleft, kright):
2  # Select the middle point
3  jmid = (jleft + jright) / 2
```

There's more on Quora...

Pick new people and topics to follow and see the best answers on Quora.

Update Your Interests

Related Questions

What is Knuth's optimization in dynamic programming?

What is the difference between dynamic programming and divide and conquer?

In an intuitive explanation what is the difference between Divide & Conquer (D & C) and Dynamic Programming (DP)?

What are the most unknown dynamic programming optimization techniques?

How important is dynamic programming for programming interviews?

Why is dynamic programming called dynamic programming?

What are differences between dynamic, divideand-conquer, and greedy programming?

What are some tricks to optimize dynamic programming solutions?

Why does divide and conquer not have an optimal substructure?

How do we determine whether to use divide & conquer or dynamic programming for certain class of problems?

+ Ask New Question

More Related Questions

Question Stats

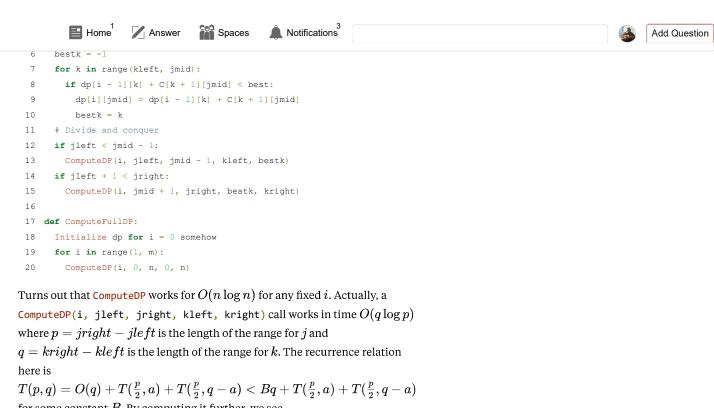
16 Public Followers

11,065 Views

Last Asked Jun 25, 2016

Edits

1 of 5



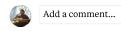
for some constant B. By computing it further, we see

$$T(p,q) < Bq + Ba + B(q-a) + T(rac{p}{4},a') + T(rac{p}{4},a-a') + T(rac{p}{4},q') + T(rac{p}{4},q-a-q') = \ = 2Bq + T(rac{p}{4},a') + T(rac{p}{4},a-a') + T(rac{p}{4},q') + T(rac{p}{4},q-a-q') < \ldots < \ < \log_2(p)Bq = O(q\log p).$$

7.7k Views · View Upvoters · Answer requested by Saad Taame

← Upvote · 81 ← Share

♦ ♦ •••



Comment Recommended All

Sponsored by Altair

Simulate more, prototype less.

Answered Jun 25, 2016

Fast and accurate engineering simulation software for better design decisionmaking.

Free trial at altair.com 000 Claudio Brandolino, Computery guy. (8)

Prelude: In a group of 100 people, find the two with the smallest square of age difference.

```
1 personOneBirthday = -1009065600;
2 // birthdays are expressed in unix timestamps set at the start of the day
3 personTwoBirthday = -722563200;
4 ageDifferenceSquare = (personOneBirthday - (personTwoBirthday))^2;
```

We are interested in finding the couple of people with the lowest ageDifferenceSquare among them.

The naive solution would be calculating it for each possible pair. We will get to our result, eventually, but it would take quadratic time (that is, for *n* points, it would take n^2 calculations).

2 of 5 11/9/2018, 7:29 AM





Add Question

To speed up the process, we divide the group between Knuth and Sussman fans. We only compare the birth months of each group's members with others in that group, and get the closest pair for each.

At the end, we compare the two groups' winning pairs (and check if there's no closest pair crossing the halves^[1]).

Now the comparisons are just $2(n/2)^2$, which is good.

If we have two cores, we can let each core take care of a group, which is better: we only have to wait for $(n/2)^2$ comparisons!

Wanna divide them further amongst Vim and Emacs supporters?

Just $4(n/4)^2$ comparisons!

More cores? $(n/4)^2$

Guys? Where are you guys? Should I just wait here while you're done?

- Core in charge of the Sussman/Vim quadrant

Incalzando: What's dynamic programming?

It's a technique we use when we have a large set of similar problems where we suspect there will be similar result: we can store each result in a data structure, so we won't have to recalculate it.

For instance, we can gradually fill up an mxm matrix where m is the number of available birthdays: the first row and column will contain a vector v, filled with all of the birthdays in our space.

At each comparison we'd check our matrix to see if the ageDifferenceSquare is already there: at m(personOneBirthday, personTwoBirthday), otherwise we swap the cell with our result.

Scherzo: Was it all wrong then? Exercises for the reader.

- Is a matrix the best data type? Won't some calculations be repeated anyway?
- Is our example's dataset worth the effort? How many people are likely to share a birthday in our dataset? What if we didn't care about the year?
- In which group would I end up? Initially I've been bold and spelled it out, which is completely backwards.
- This was a huge waste of time complexity wise time wise. Would it have been the same if we were looking for the two people with the closest sum of ageDifferenceSquare between each one and one fixed date?

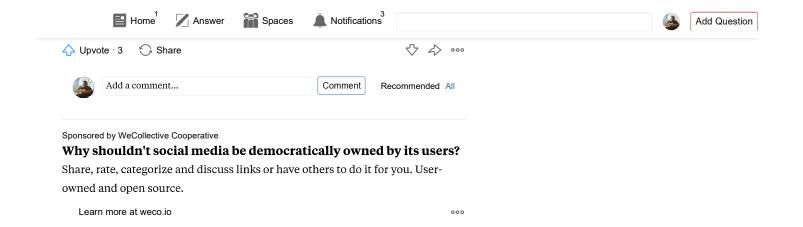
Footnotes

[1] A Divide-and-Conquer Approach

3.3k Views · View Upvoters

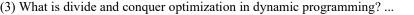
Your feedback is private.

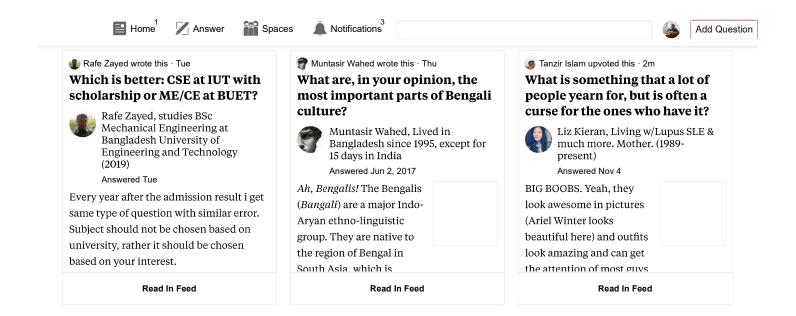
3 of 5 11/9/2018, 7:29 AM (3) What is divide and conquer optimization in dynamic programming? ... https://www.quora.com/What-is-divide-and-conquer-optimization-in-dy...



4 of 5

https://www.quora.com/What-is-divide-and-conquer-optimization-in-dy...





5 of 5