Hash Code - Discussion

ACM club meeting: Feb 26, 2020

Problem Statement

Download

https://hashcodejudge.withgoogle.com/#/rounds/6313004828196864/submissions/

Problem Statement

You want to scan a lot of books over the course of a fixed number of days.

You have a lot of libraries, which you can sign up one at a time. Each library takes some time to sign up. Once this process is done, that library can scan books indefinitely at some rate.

Libraries can only scan books that they own, and some books are more important to scan than others. Each distinct scanned book gives some score upon being scanned.

Choose the order to sign up libraries and scan books so as to maximize the score from scanned books. (On some fixed data sets)

Solution Ideas

Problem is far too complicated to hope for an efficient exact solution.

A common strategy was to separate the problem into 2 parts: Choosing an order to sign the libraries up in, and then choosing the books that the libraries send.

For the given datasets, the first part is far more important to do well.

Choosing a signup order

Greedy strategies were very common and worked very well, a good implementation of greedy can take you to the top of the leaderboard alone.

General strategy

For each library, create some heuristic to say how good it would be to sign the library up. Then sign up the libraries, ordered by this heuristic.

Recall that each library has the following attributes:
1. The set of books, each of which has a value (different libraries may have the same book)

2. The time it takes to sign up the library so it can start scanning books

3. The rate at which the library can scan these books

Can you come up with a decent heuristic?

Heuristic Example

Higher value of the heuristic = better

Heuristic: Sort the books in increasing order of value, then sum up the books that we can scan from now until the end of time with this library. Divide by the time it takes to sign up that library.

Choosing the books each library scans

Just go through each library from most valuable book to least, and scans books that haven’t been sent yet (keep track with a set).

Works pretty well, combined with the previous strategy you can probably get top 15% or so.

Simple optimizations

You can make a local solver to see the number of points your solution gets and us it to refine your greedy strategy, multiply/add by random constants and see how it changes!

Also, add some randomization to your greedy strategy and run the algorithm a bunch on your computer and your teammates computers, taking the best score.

Alternatively, stick with a given correct solution, perform swaps of the library ordering, and calculate the new score. If it’s an improvement, keep it and continue iterating this on your computer! You can use your teammates for each test case ;P

Choosing the books each library scans: Part 2

We can create a bipartite graph from libraries to the books they can scan.

Then this is a maximum weight vertex matching problem, if we add dummy vertices as needed. We can then run the hungarian algorithm to optimize, though it's probably too slow [O(V^3)]. Maybe there's a better way to interpret this.

https://codeforces.com/blog/entry/73710?#comment-582254

Choosing the books each library scans: Part 2

We can create a bipartite graph from libraries to the books they can scan. If we just want to optimize the number of books scanned:

Then from the source, there is an edge to each library with capacity equal to the number of books said library can scan.

Each book has a weight-1 edge to the sink. The max flow in this graph is the maximum number of books we can scan, and we can check all the edges to get an optimal answer, though it doesn’t take into account the number of edges.

Choosing the books each library scans: Part 2

To try to take cost into account, we can choose augmenting paths through our flow graph that improve the score. Then we have an optimal answer when we have no more augmenting paths, and we can terminate early if it takes too long.

Further optimizations

We can make some more optimizations, but recall that we are given fixed datasets.

It’s important to check out the data sets and see what kinds of specific problems we are dealing with.

Given the following datasets, can you come up with any ideas that might be good for that specific situation?

Datasets

Dataset B: All book values the same, and once signed, every library can scan distinct books one per day until the end of time. (Only variable is library sign-up time)

Dataset C: Libraries can scan all books instantly once signed up

Dataset D: Libraries have the same sign up time and can scan their books almost instantly, all book values are the same. Each book has the same value and is at no more than 3 libraries.

Dataset E: Only around 200 days until the end of time (Can only sign about 80 libraries)

Dataset F: Pretty big and pretty hard

Dataset C

Dataset C: Libraries can scan all books instantly once signed up

This is an instance of the set-cover problem, but there are more weights. We can solve it with integer linear programming by modifying the formulation on the wikipedia page.

https://en.wikipedia.org/wiki/Set_cover_problem

From https://codeforces.com/blog/entry/73710?#comment-582226, they scored very well with this.

Dataset D

Dataset D: Libraries have the same sign up time and can scan their books almost instantly, all book values are the same. Each book has the same value and is at no more than 3 libraries.

We need to choose a subset of the libraries which contains a maximal number of distinct books. If we want to scan each book, then our goal is to sign up one of the 3 libraries that contain it. This can be viewed as trying to solve the maximum satisfiability problem, related to 3-SAT.

-This strategy was a part of the winning solution:
https://codeforces.com/blog/entry/73710?#comment-582218

SAT reduction example

Library 1 can scan books 1, 2, 3

Library 2 can scan books 1, 2, 4

Library 3 can scan books 2, 3, 4

We can scan all a book for each of the following true clauses (numbers represent if the ith library scans stuff):

(book 1) and (book 2 ?) and (book 3) and (book 4)

(1 or 2) and (1 or 2 or 3) and (1 or 3) and (2 or 3)

Dataset E and F

Dataset E: Only around 200 days until the end of time (Can only sign about 80 libraries)

Dataset F: Pretty big and pretty hard

Uh oh… actually, the local optimization we talked about in the greedy solution probably works quite well on these, especially E. (Untested)

Questions/Comments?