Google Hashcode 2016: delivery

Звіт підготували Олександр Потапов та Бриляк Павло

Дизайн рішення

Перед розробкою нашого алгоритму, ми зробили декілька фундаментальних рішення:

- Ми не пересуватимемо товари з складу на склад, навпаки всі замовлення будуть доставлятися напряму зі складу, на якому є необхідні компоненти замовлення.
- Дрон не виконуватиме більше ніж одне замовлення за раз, адже виходячи з вхідних данних на виконання одного замовлення потрібно в середньому більше одного дрона.
- Замовлення не почне виконуватись поки не буде достатня кількість необхідних вільних дронів, для його виконання.

Головна ідея алгоритму полягає в еврестичному знаходження такого замовлення в момент часу t, яке можна виконати найшвидше(earliest-completion-first).

На кожному кроці ми визначаємо замовлення, яке може бути виконане найшвидше, для цього використовуємо відповідну стратегію:

1. Зафіксуємо замовлення O.
2. Проходимось по кожному складу й фіксуємо всі наявні елементи на складі, які є підчастинами нашого замовлення, формуючи пари *(склад, товари).*
3. Розбиваємо пари на менші пари, якщо вага товарів більша ніж вантажопід'ємність дрона.
4. Сортуємо пари відповідно до відстані між складом й замовленням від найближчого до найдальшого.
5. Знаходимо для кожної пари *(склад, товари)* вільного дрона, формуючи пари *(дрон, склад)* жадібним алгоритмом мінімізуючи значення

   відстань(дрон, склад) + відстань(склад, замовлення)

Опис й аналіз алгоритмів

Основну роботу виконують два алгоритми:

makePacks(order) – розбиває замовлення на ключ-значення, утворюючи пари *(склад, товари),* де склад – це координати сховища, а товари – всі

частини, які присутні на цьому складі і є в замовленні. Сортує ці пари за віддаленням складу від замовлення.

Приклад роботи:
Вхід*:* замовлення у вигляді пари *(координати, товари)* –
(370, 341), [271, 314, 26, 32].
Вихід: список пар *(склад, товари)* відсортований за віддаленням від замовлення -
{(360, 340): [271, 26, 314], (318, 320): [314], (210, 190): [32, 26,271]}

Алгоритм має складність – O(pn)
p – кількість елементів в замовленні,
n – кількість сховищ,
адже до самих елементів на складах ми доступаємось за індексом, що має складність O(1).

greedy_algorithm(packs) – еврестичний алгоритм, який підбирає дронів відповідно до їх локації для кожної пари *(склад, товари)*, утвореної попереднім алгоритмом makePacks(). Основне завдання алгоритму - знайти мінімальне значення виразу

відстань(дрон, склад) + відстань(склад, замовлення)

Приклад роботи:
Вхід: список пар *(склад, товари)* –
{(360, 340): [271, 26, 314], (318, 320): [314], (210, 190): [32, 26,271]}
Вихід: список трійок *(дрон, склад, товари)* –
[
(drone_1, (360, 340), [271, 26, 314]),
(drone_2, (210, 190), [32])
]

Де drone_1 – найближчий вільний дрон до складу (360, 340), drone_2 – до складу (210, 190).

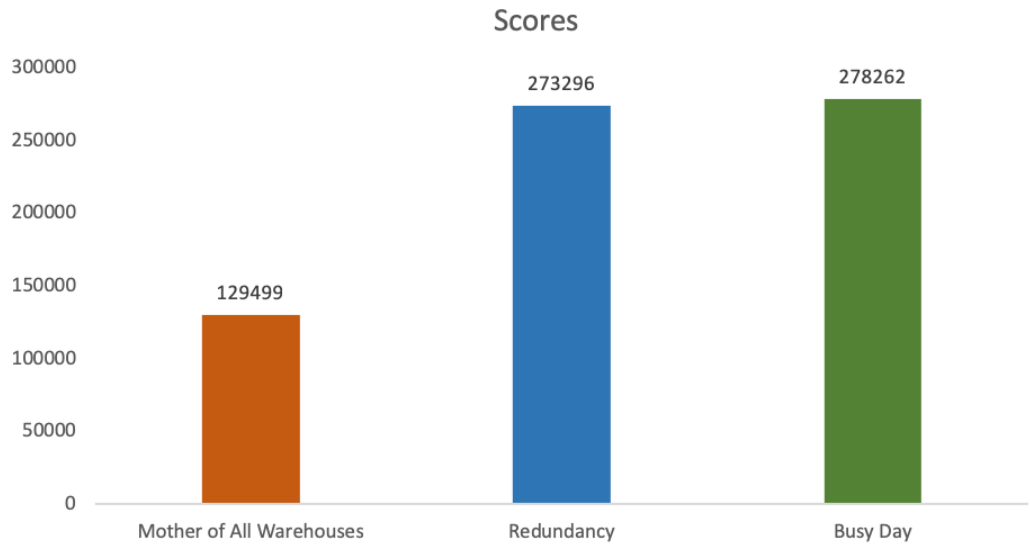
Алгоритм має складність – O(pkn)
p – кількість підчастин замовлення
k – кількість складів з наявними частинами підзамовлення
n – кількісіт вільних дроні

Результати

Ми протестували наші алгоритми на трьох типах вхідних данних, результати ми можемо бачити на мал. 1. Найменший результат ми маємо на найменшій за площею карті з одним складом, 20 дронами та найменщрю кількістю замовлень – 800 – Mother of All Warehouses. Більший результат маємо на більшій карті - Redundancy. Відповідно найбільший результат маємо на найбільшій карті, з найбільшою кількістю замовлень та складів – Busy Day. Такий результат є очікуваний, оскільки наші алгоритм підбирають пари відстаней *(склад, замовлення)* та *(дрон, склад)* з найменшим значенням їх суми. Зі збільшенням кількості складів, замовлень та дронів алгоритм підбирає кращий розв'язок, а більша кількість замовлень дає змогу набрати більше балів.

Mother of All Warehouses		Redundancy		Busy Day	
20 drones, 800 orders, 1 warehouse		30 drones, 1000 orders, 10 warehouses		30 drones, 1250 orders, 16 warehouses	
map	240x400	map	300x500	map	400x600
drone	score	drone	score	drone	score
1	5969	1	9597	1	9647
2	6341	2	9173	2	9159
3	6951	3	9780	3	9822
4	6235	4	9021	4	9310
5	6732	5	9266	5	9628
6	6035	6	9484	6	9210
7	7206	7	9649	7	9101
8	7135	8	9465	8	9333
9	6596	9	10107	9	10150
10	6935	10	9421	10	9363
11	6742	11	9314	11	8859
12	6204	12	9031	12	8809
13	6298	13	10058	13	8486
14	6118	14	8859	14	9033
15	6391	15	8913	15	8628
16	6059	16	8860	16	9143
17	6330	17	8553	17	9503
18	6772	18	9067	18	9503
19	6143	19	9029	19	10109
20	6307	20	8933	20	9966
-	-	21	8974	21	9598
-	-	22	8779	22	10255
-		23	8576	23	9536
-	-	24	8759	24	9907
-	-	25	9160	25	9028
-	-	26	8558	26	8645
-	-	27	8669	27	8978
-	-	28	8752	28	8785
-	-	29	9012	29	8469
-	-	30	8477	30	8299
total score	129499	total score	273296	total score	278262

мал. 1



мал. 2

Загальні результати ми можемо бачити на мал. 2.

Висновки

Рішення, які ми прийняли при розробці алгоритму(див. Дизайн рішення) значно спростили нам вирішення завдання, отже наш жадібний алгоритм знаходить оптимальне рішення просто перебираючи всі можливі варіанти, щоправда має складність роботи близьку до O(n^3), n завжди доволі мале число. Ми могли б поексперемнтувати з іншими алгоритмами й зробити порівняння, але для цього нам би довелось повністю змінювати підхід до вирішення завдання(робити кластеризацію замовлень, мережу взаємодії між дронами чи щось інше). Наш підхід показав гідний результат, який прирівнюється до 18 місця в загальному рейтингу на найбільшій карті - BusyDay.

Посилання

Глобальний рейтинг - https://codingcompetitions.withgoogle.com/hashcode/archive/2016
GitHub – https://github.com/PavloBryliak/AdFontes