Софийски университет „Св. Климент Охридски“
Факултет по Математика и Информатика

Проект

по

Структури от Данни и програмиране

УЧЕБНА ГОДИНА 2023/2023

Изготвил: Радослав Каратанев

Факултетен номер: 3MI0800036

Група: 4

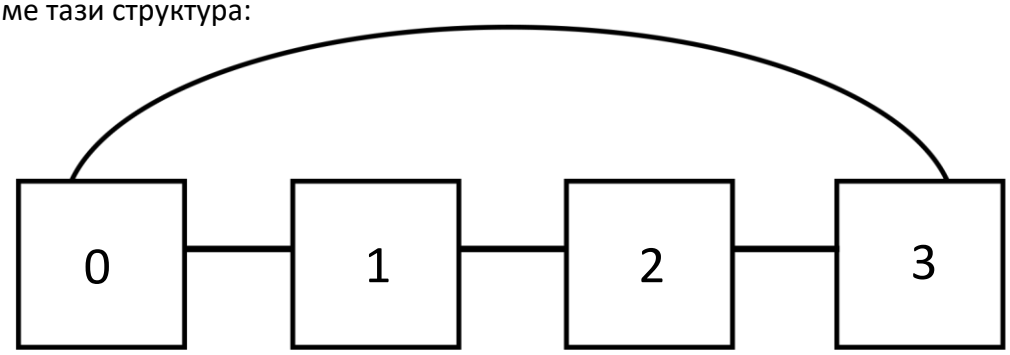
Съдържание:

09.01.2023

София

Намиране на оптимален път в свързан списък

Като започнем от горе надолу, в main.cpp на папката “A)”, структурата skipList е нашият модифициран свързан списък. SkipList* next е следващата гара в пътя на нашия пътнически влак, а SkipList* skip е следващата гара на експресния влак. Ако например имаме тази структура:



Тук Node(0)->next = Node(1) и Node(0)->skip = Node(3).

Функциите: 1) toList – приема като параметри масив и размер на масива и връща свързан списък от елементите на този масив.

2) makeSkips – приема свързан списък и му прави skip-овете

3) deleteList/printList – трие паметта/принтира свързан списък

4) Journey – Алгоритъмът, който изчислява оптималия път, който можем да поемем. Приема свързан списък от вид SkipList(с направени skip-ове) и масив от имена на гари, които задължително искаме да посетим. Алгоритъмът работи както следва:

А) Ако гарата на която сме вмомента е една от тези, които искаме да посетим задължително, то в маршрута ни добавя сегашната гара.

Б) В противен случай, ако сегашната ни гара има skip, който води директно до друга гара, която искаме да посетим, директно skip-ва до нея.

В) В противен случай, ако сегашната ни гара има skip, проверяваме дали има една от желаните гари между сегашната и тази, до която имаме skip. Ако има, преминаваме на next гарата, а ако няма, преминаваме на skip гарата.

Този алгоритъм продължава да изчислява маршрута докато няма следваща гара, която бихме могли да достигнем.

09.01.2023

София

Намиране на оптимален път в граф

Отново започвайки от горе надолу, в main.cpp на папката “B)”, структурата Node представлява списъкът, който се връща като краен резултат от алгоритъма. Структурата Pair свърза дадена локация в графа с съответен индекс. Структурата Edge представлява ребро на графа.

Функциите: 1) isContained – връща дали в даден масив от тип Pair се съдържа съответен елемент на графа.

2) printList/deleteList – принтират/трият паметта на свързан списък от тип Node.

3) algorithm - Това е функцията, която ни пресмята пътя. Функцията се разделя на 2 главни части:

А) Създаване на граф от подадения като параметър файл

Б) Намиране на оптималния път в този граф

Графът ни е представен като масив от вектори от тип Edge. Всичките данни от файла се вкарват в 3 опашки “froms”, “tos” и “times”. От тези 3 опашки последователно се добавят в графа ни.

След като е създаден напълно графът, използваме алгоритъмът на Дийкстра за да намерим колко (като време) е най-краткият път от “Railstation” до всяка друга точка в графа.

Последната стъпка на алгоритъма е: 1) Гледа с кои точки в графа е свързана текущата

2) Движи се по тази връзка, която е с най-малко време

3) Повтаря тези стъпки, като на всяка стъпка проверява дали ако се придвижи още веднъж ще има достатъчно време да се завърне до началната си точка.

09.01.2023

София

Оптимизиране на използвано място за сувенири

Отново започвайки от горе надолу в main.cpp в папката “C)”, класът Box е основният ни клас за положената ни задача. Той съдържа “name”, което представлява името на текущата кутия, numberOfChildren, което описва броя други кутии, който се съдържат в текущата. Също съдържа вектор от тип Box* children, който представлява списък на всички кутии, който се намират в текущата (още нейните „деца“). Последно има вектор от тип std::string, който представлява списък от предметите, намиращи се в кутията. Най-важните фуцнкии на този клас са: 1) addChild – добавя кутия в текущата

2) (!) removeChild – приема като параметър индекс и премахва кутията с този индекс от текущата

Функцията minimizeBoxes е алгоритъмът за нашата задача. Отново можем да го разделим на няколко основни части: 1) Чете подаденият на функцията файл и създава всички нужни кутии, но без да слага една кутия в друга.

2) Когато има всички кутии създадени, започва да влага кутии една в друга(с addChild функцията)

3) След изпълнението на първите две стъпки, преглежда кутиите за такива, който не са нужни. С функциите removeChild и addChild преобразува структурата ни в такава, която няма ненужни кутии.

Когато алгоритъмът премине през стъпка 3) без да направи нито една промяна, то нашата крайна структура е създадена.

Допълнителни ресурси

Всичко е периодично качвано в GitHub, като за момента репото е Private.

https://github.com/radkotoktv/SDP-Project

09.01.2023

София