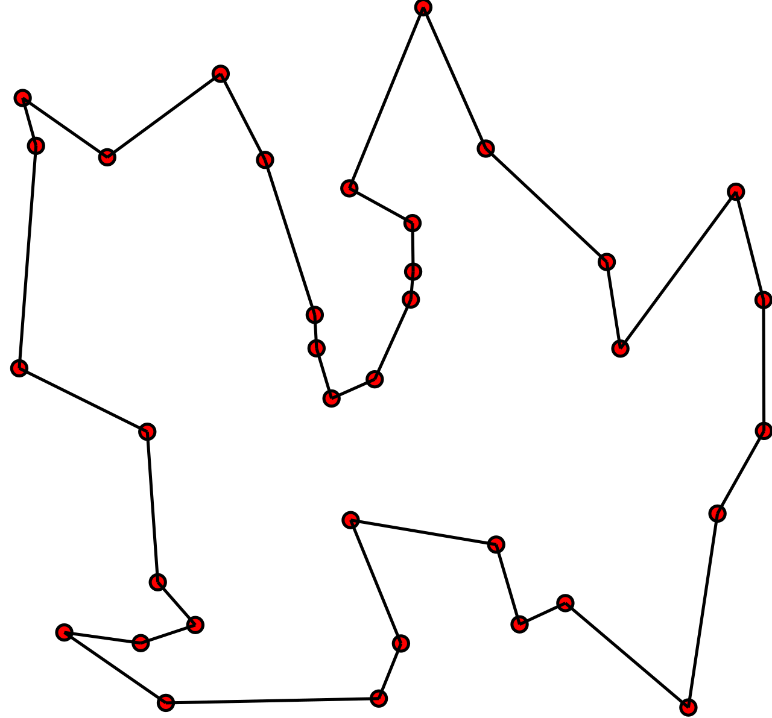
*УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ*

*ФАКУЛТЕТ „ИНФОРМАЦИОННИ НАУКИ“*

*Курс „Основи на програмирането“*

Анализ, решение и имплементация на задача „Бензиностанции“

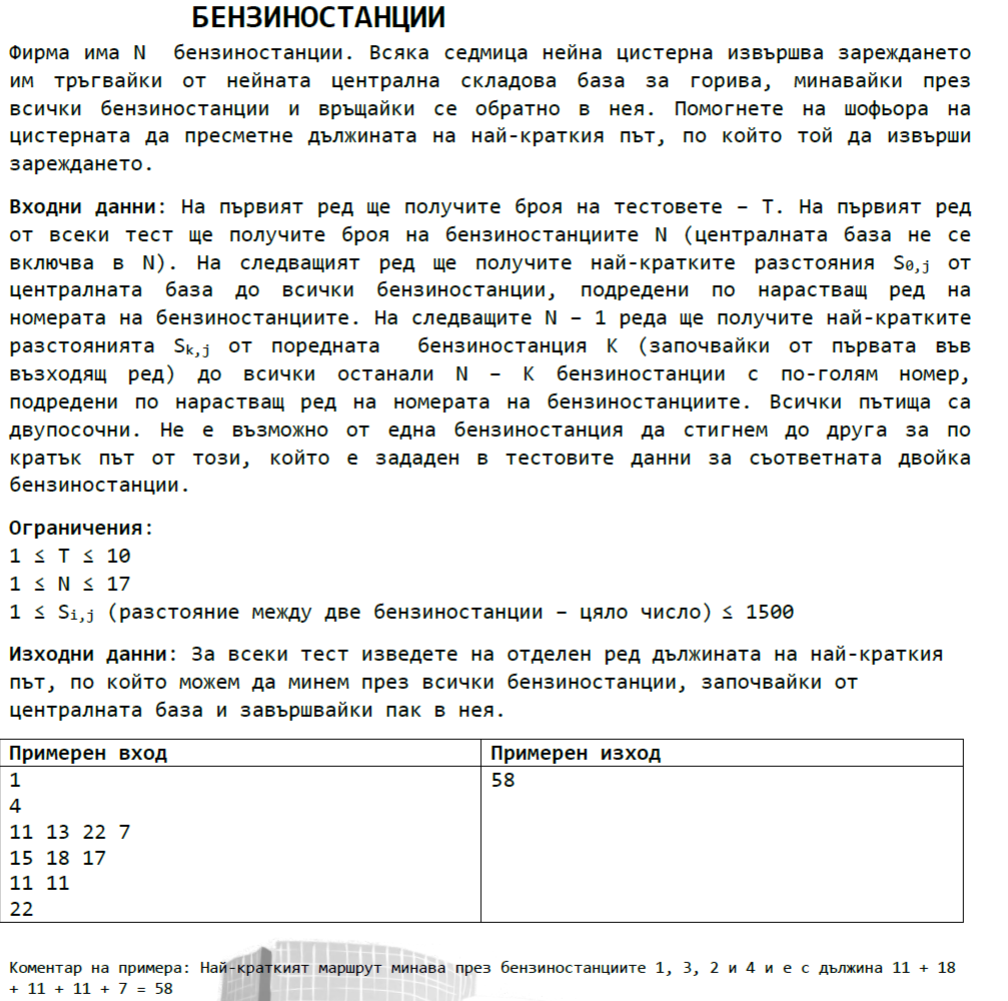
*Автор: Костадин Динков, 1-ви курс, направление ИКН, фак. № 46355з*



# Абстракт

Целта на този документ е да представи практическа имплементация на решение на задачата с „Бензиностанциите“, зададена като освобождаваща от изпит в курса „Основи на Програмирането“ ( УНИБИТ, ИКН, първи семестър) и по този начин да запознае читателя с един от най-популярните оптимизационни проблеми в информатиката и компютърните науки. Проблема се анализира и се разглеждат няколко възможни подхода за решаването му, като метод с изчерпателно търсене и приблизителни решения. За самата имплементация на решението се използва алгоритъма на **Белман-Хелд-Карп** за класическия проблем с пътуващият търговец (**Traveling Salesman Problem**). Обясненията са насочени към хора с начален опит в програмирането и алгоритмите, но ще бъдат най-подходящи за тези, които сами са се опитали да се справят със задачата. Решението е имплементирано на C#, използвайки .Net Core технологията.

# 1. Условие На Задачата

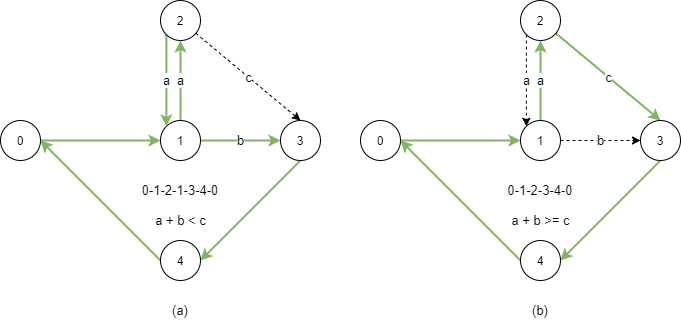


# 2. Анализ

От условието на задачата е видимо, че тя е много прилича на класическата задача за пътуващия търговец (Traveling Salesman Problem – TSP). Като входен параметър имаме пълен, неориентиран граф. Пълен, защото имаме пътища (ребра) от една бензиностанция (връх) до всяка друга бензиностанция. Трябва да намерим най-краткия път от даден начален връх , така че този път да премине през всички върхове в графа и да се върне обратно в първия. В теория на графите, затворен цикъл, който преминава през всички върхове точно веднъж се нарича Хамилтонов цикъл. В този документ, за по-кратко, такъв път ще наричаме **обиколка**.

## 2.1. С повторение или без повторение

За разлика от класическия TSP, в който имаме ограничението, че трябва да посетим всеки връх веднъж и само веднъж, нашата задача няма такова условие. Т.е. е възможно да имаме оптимален път, който посещава една бензиностанция повече от веднъж, като например 0-1-2-1-3-4. Това означава, че е пътят от 2-3 е по-дълъг отколкото сумата на пътищата от 2-1 и 1-3, така че за цистерната е по-добре да се върне от 2 до 1 и след това да стигне от 1 до 3, отколкото да стигне от 2 директно до 3. Това е демонстрирано на Фигура 1 (а). Но в нашият вариант на задачата имаме и допълнителното ограничение, че „не е възможно от една бензиностанция да стигнем до друга за по-кратък път от този, който е зададен в тестовите данни за дадената двойка“. Това означава, че примера по-горе не отговаря на условията, т.е. директният път от 2 до 3 винаги ще е по-кратък или равен на сумата на пътищата от 2-1 и 1-3 (Фигура 1(b)).



Фигура 1   
(а) Възможна обиколка, когато можем да повтаряме посещения на върхове.  
(b) Възможна обиколка, когато можем да повтаряме върхове и не съществува по-кратък път от директния между два върха.

От тук следва и е лесно доказуемо, че ако в нашата задача имаме оптимална обиколка с повторения, като например

където са върхове от графа, а е разстоянието между два върха, то ако премахнем повторенията на , ще получим обиколка

който е по-кратък или равен на първия, защото

Всъщност ще можем да премахнем повторенията на всеки връх и да получим обиколка без повторения, която е с по-малка или равна дължина на тази с повторенията. Единственото условие е върхът, който сме избрали за начало да се повтаря в края на обиколката, но можем да премахнем други негови повторения в средата. Като обобщение можем да кажем, че ако е дължината на оптимална обиколка, в която има повторения на върхове, а е дължината на такава, в която няма повторения то:

Какво означава този извод за нашия проблем? Това, че в условието не е дадено ограничение за броя на посещенията на един връх няма значение за крайният ни резултат. Обиколката без повторения винаги ще е по-кратка или равна на такава с повторения.

## 2.2. Извод

От написаното до тук следва, че можем да приравним нашата задача на класически частен случай на TSP, а именно MTSP (metric TSP) където е валидно правилото за неравенства в триъгълника

**Дефиниция 2.1.** Неравенства в триъгълника :

където e функция за разстояние между два върха.

Вече имаме насока, в която можем да направим проучванията си за намиране на решение на проблема. От една страна, TSP e един от най-изследваните и популярни проблеми в компютърните науки и за него съществуват различни решения и алгоритми. От друга страна, към датата на писането на този документ, не е известно бързо решение което да дава точен резултат. Под бързо се има предвид решение, което да дава резултат в полиномно време, т.е. времето за изпълнение е полиномна функция на размера на входните данни. Оказва се, че времето за намиране на точен резултат на нашият проблем не е полиномна, а експоненциална функция на броя на бензиностанциите. За да припомним, ако n е броя на бензиностанциите, то полиномна функция е например , или. Експоненциалната функция може да изглежда така .

Това което ни остава е да намерим най-подходящ алгоритъм, който да ни дава точен резултат при максимум 17 бензиностанции.

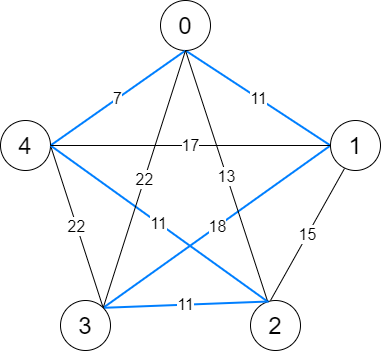
# 3. Алгоритъм с Изчерпателно Търсене

Най-интуитивният начин за решение е да атакуваме проблема директно, чрез използването на brute-force (груба сила?) алгоритъм, т.е. като изброим (енумерираме) всички възможни обиколки, изчислим техните дължини и накрая изберем като резултат най-малката от тях. Ако приемем, че броят на градовете е N, то всички възможни обиколки от даден връх в графа са (факториел), a **различимите** са . Това е така защото една обиколка може да бъде направаена и в обратна посока.

|  |  |  |
| --- | --- | --- |
| Обиколка | Обиколка в обратна посока | Дължина |
| 0-1-2-3-4-0 | 0-4-3-2-1-0 | 11+15+11+22+7=66 |
| 0-1-3-2-4-0 | 0-4-2-3-1-0 | 11+18+11+11+7=58 |
| 0-1-3-4-2-0 | 0-2-4-3-1-0 | 11+18+22+11+13=75 |
| 0-1-2-4-3-0 | 0-3-4-2-1-0 | 11+15+11+22+13=72 |
| 0-1-4-3-2-0 | 0-2-3-4-1-0 | 11+17+22+11+13=74 |
| 0-1-4-2-3-0 | 0-3-2-4-1-0 | 11+17+11+11+22=72 |
| 0-2-1-3-4-0 | 0-4-3-1-2-0 | 13+15+18+22+7=75 |
| 0-2-1-4-3-0 | 0-3-4-1-2-0 | 13+15+17+22+22=89 |
| 0-2-3-1-4-0 | 0-4-1-3-2-0 | 13+11+18+17+7=66 |
| 0-2-4-1-3-0 | 0-3-1-4-2-0 | 13+11+17+18+22=81 |
| 0-3-1-2-4-0 | 0-4-2-1-3-0 | 22+18+15+11+7=73 |
| 0-3-2-1-4-0 | 0-4-1-2-3-0 | 22+11+15+17+7=72 |

Фигура 2.

Диаграма на граф с 5 върха и оптимална обиколка 0-1-3-2-4-0 с дължина 58. В таблицата са дадени всички възможни обиколки с начален връх 0, както и техните дължини.



На Фигура 2 е дадена диаграмата на примерния входен тест от заданието на проблема. В таблицата на Фигура 2 са изброени всички възможни пътища с начален връх 0, преминаващи през всички останали върхове и връщащи се обратно до началния връх. Виждаме, че броят на възможните обиколки е 24, което е точно :

Виждаме също, че една обиколка може да бъде направена и в обратна посока и това няма да промени нейната дължина. Например обиколката 0-1-2-3-4-0 с размер 66 е с еднаква дължина и в обратна посока 0-4-3-2-1-0. Това намалява размера на възможните решения на 12, т.е. 2 пъти:

Факториела е експоненциална функция, която нараства изключително бързо. В Таблица 1 е показано как броят на възможните обиколки расте с увеличаване на броя на бензиностанциите.

|  |  |  |
| --- | --- | --- |
| Брой на бензиностанциите | Брой на възможните обиколки | Време за изчисление при 15000000 операции/сек |
| 2 |  |  |
| 3 |  |  |
| 4 |  | 0.0004 мс |
| 5 |  | 0.0016 мс |
| 10 |  | 24.192 мс |
| 15 |  | 1.6 часа |
| 18 |  | 274 дни |
| 20 |  | 260 години |
| 21 |  | 5 200 години |
| 22 |  | 109 202 години |
| 23 |  | 2 402 450 години |

От Таблица 1 виждаме, че ако получим като входни данни 18 бензиностанции (17 + началната), което е максимумът по условие, ще имаме възможни обиколки! По време на писането на този документ, моят компютър успява да направи изчисление за 15 000 000 обиколки за 1 сек. Това означава, че **(18-1)! ще отнеме около** **274 дни**. Увеличаването със само още 1 бензиностанции т.е (**19-1)! ще отнеме 13 години**, а (20-1)! – 5200г.

Таблица 1.

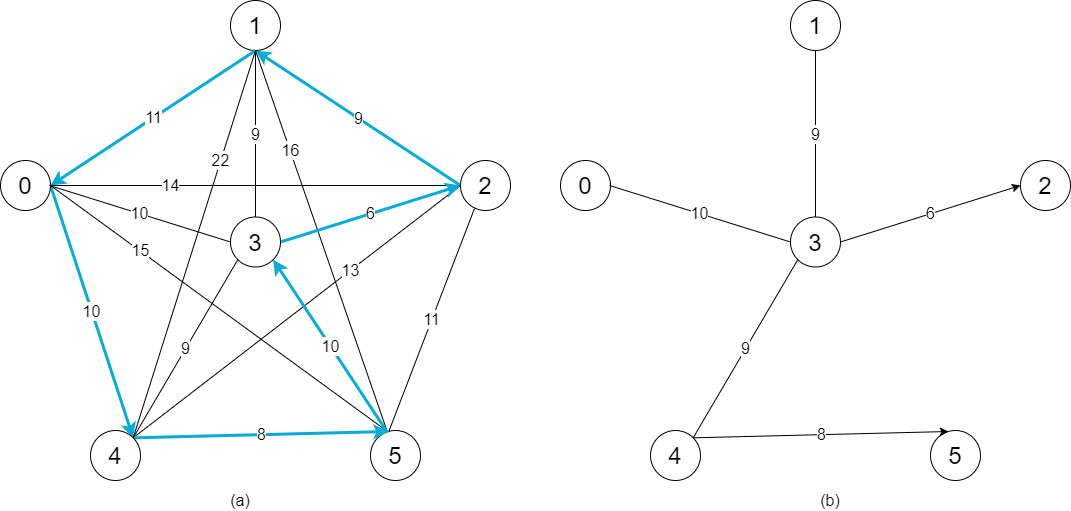
Нарастване на възможните обиколки според броя на бензиностанциите

Въпреки че в условието на задачата не е дадено крайно време за намиране на резултат, можем смело да предположим, че търпението на проверяващият е с горна граница от около 7 сек.

# 4. Алгоритъм за Приблизително Решение

Нека разгледаме диаграмата граф (Фигура 3a), който отговаря на условията на нашата задача. Виждаме, че има път от всеки връх, до всеки друг връх, разстоянията между върховете са положителни и отговарят на правилото за неравенства в триъгълника. Нека OPT е дължината на оптималната обиколка с начало връх 0 и път 0-4-5-3-2-1-0. OPT = 54. По дефиниция знаем, че ако построим минимално покриващо дърво върху началния граф (Фигура 3b), сумата на ребрата на това дърво ще е минимална т.е. гарантирано е че сумата на разстоянията между бензиностанциите ще е минималната. Нека означим тази сума като MST.

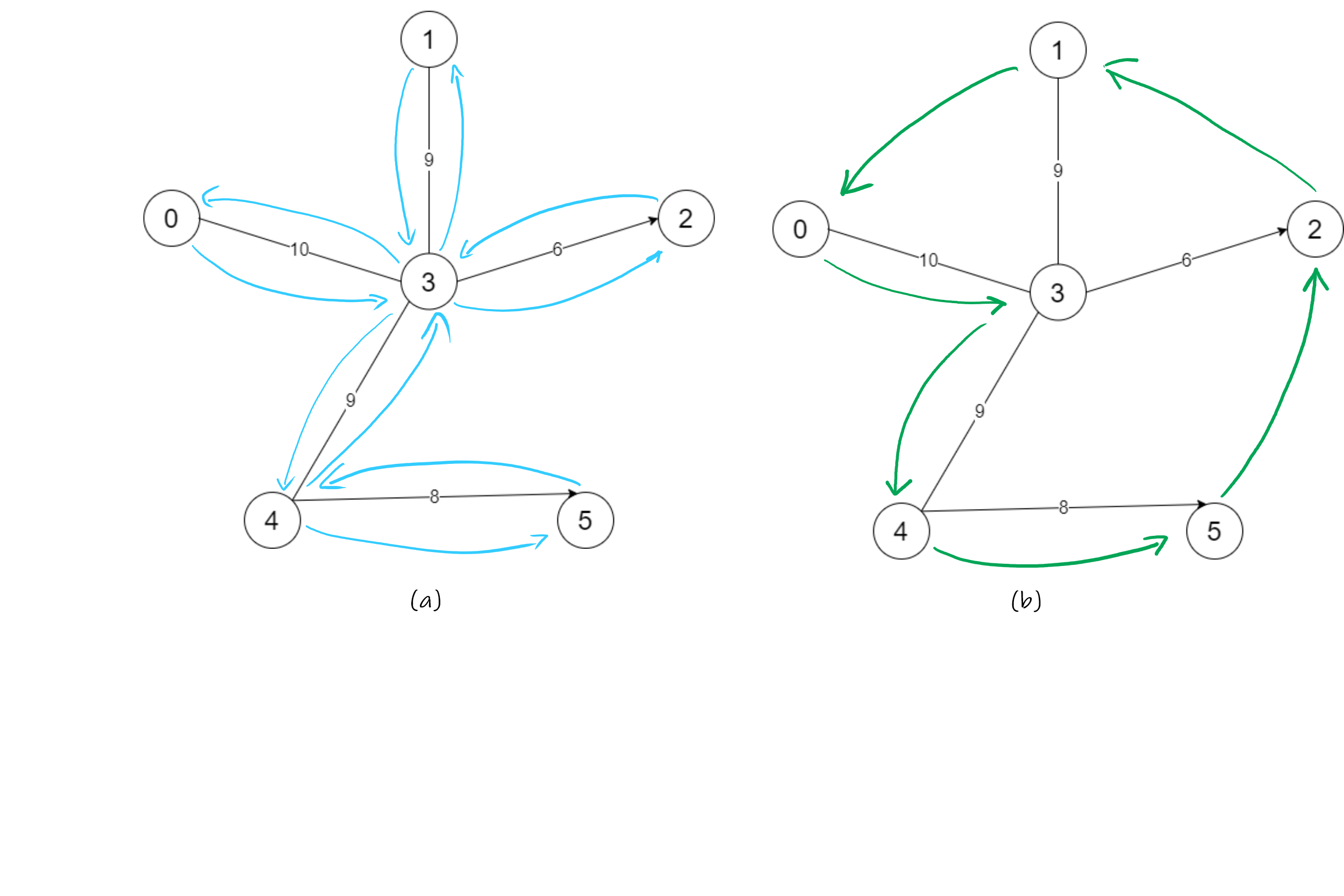
Фигура 3  
(а) Пълен граф, с разстояния отговарящи на правилото за неравенства в триъгълник.  
(b) Минимално покриващо дърво на същия граф.



**Твърдение 4.1**:

**Доказателство** чрез противоречие**:** Нека приемем случай на граф, при който , което означава, че дължината на оптималната обиколка е по-малка от сумата на ребрата на минималното покриващо дърво. Ако премахнем едно ребро от ще получим ацикличен граф, който включва всички върхове точно веднъж, следователно той е покриващо дърво със сума на ребрата . Получава се, че , следователно не е минимално – противоречие.

Ако направим търсене в дълбочина (DFS) върху MST, преминавайки през всяко ребро точно по 2 пъти (Фигура 4a) ще получим обиколка PT с дължина 2 пъти по-голяма от тази на MST:



Фигура 4  
Със сини стрелки е демонстрирано обхождането на минималното покриващо дърво чрез търсене в дълбочина (DFS)

Записваме всеки връх когато го посещаваме, чрез DFS алгоритъма и за PT получаваме:

Използвайки това, че началният ни граф отговаря на неравенствата в триъгълника (Дефиниция 2.1), можем да премахнем всяко последващо посещение на връх в пътят на PT и ще получим нов тур :

Следователно:

Като заключение можем да твърдим, че резултатът от описаните стъпки по-горе ще ни гарантира краен резултат, който в най-лошият случай ще е 2 пъти по-голям от оптималния. Самият алгоритъм можем да обобщим така:

.

1. Намираме минимално покриващо дърво
2. Обхождаме това дърво чрез търсене в дълбочина и записваме всеки връх по реда на посещението му в поредица.
3. Обхождайки поредицата от ляво надясно, премахваме всеки връх, който вече е бил записан поне веднъж.
4. В края на поредицата добавяме началния връх.

|  |  |  |
| --- | --- | --- |
| Брой на бензиностанциите | Сложност по време при brute force | Сложност по време при MST+DFS |
| 2 |  | 4 |
| 3 |  | 9 |
| 4 |  | *16* |
| 5 |  | *25* |
| 10 |  | *100* |
| 15 |  | *225* |
| 18 |  | 289 |
| 20 |  | 400 |

В Таблица 2 може да се види приблизително сравнение на времевата сложност (time complexity) между алгоритъма чрез изчерпателно търсене(Глава 3) и алгоритъма за приблизителен резултат разгледан тук. Виждаме, че в сравнение с brute-force алгоритъма, MST+DFS работи светкавично, въпреки че е с квадратно време .

Таблица 2  
Приблизително сравнение на времевата сложност (time complexity) на двата алгоритъма разгледани до тук.

## 4.1 Извод

За съжаление, както се вижда и от заглавието на текущата глава, бързият алгоритъм е неточен. Дава ни приблизителна стойност за най-краткия път. Единствената гаранция е, че резултатът от ще бъде максимум 2 пъти по-голям от дължината на оптималния път. От експериментални данни знаем, че оптималната обиколка за примерния граф е 54, а пътят минава през 0-4-5-4-3-2-1-0. Алгоритъмът, който описахме изчислява обиколка с дължина 58 и път 0-3-4-2-5-1-0. В практиката съществува и по-точен алгоритъм – този на Christofides, който гарантира, че в най-лошият случай, резултатът ще е 1.5 пъти по-голям от оптималния, което е значително подобрение. Нито един от приблизителните алгоритми не е решение на нашата задача. Трябва да получим точен резултат, който лесно да може да бъде тестван за коректност.

# 5. Bellman-Held-Karp

Съществуват много публикации, изследвания и литература по проблема за „пътуващия търговец“ (TSP). Както беше споменато в Глава2, TSP се класифицира като NP-hard проблем. Опростено казано, не можем да изчислим точен резултат в полиномно време. От една страна, ако искаме точно решение, ще трябва да жертваме време за изпълнение. От друга, ако искаме бързо решение, ще трябва да жертваме точност на резултата. От условието на задачата е видимо, че търсим точен резултат. Разбрахме също, че чрез метода на изчерпателно търсене ще са ни нужни около 11 години за да изчислим точен резултат при брой на бензиностанциите от 17. Остава въпросът „Можем ли да се справим по-добре“.

През 1962 година, Ричард Белман от една страна и Майкъл Хелд и Ричард Карп от друга, независимо едни от други предлагат алгоритъм за решение на TSP, който използва техника наречена динамично програмиране и постига сложност по време . В Таблица 3 е направено сравнение на броя операции при алгоритъм с изчерпателно търсене и алгоритъма на Bellman-Held-Karp. Въпреки, че алгоритъма на Хелд-Карп също е с експоненциална сложност по време, от таблицата е видимо, че работи много, много по-бързо. Като използваме данните, че съвременен компютър може да извършва изчисления за 15 000 000 обиколки в секунда (грубо) получаваме, че за 18 бензиностанции ще са ни нужни 262 милисекунди, ако използваме алгоритъма на Хелд-Карп.

|  |  |  |
| --- | --- | --- |
| Брой на бензиностанциите | Време за изпълнение при brute force | Време за изпълнение при Bellman-Held-Karp |
| 5 | 0.0016 мс | 0.001 мс |
| 10 | 24.192 мс | 0.239 мс |
| 15 | 1.6 часа | 21.3 мс |
| 18 | 274 дни | 262 мс |
| 20 | 260 години | 1.34 сек |
| 21 | 5200 години | 3 сек |
| 22 | 109202 години | 6.6 сек |
| 23 | 2402450 години | 15 сек |

Таблица 3  
Приблизително сравнение на времето за изчисление на резултат при норма от 15млн. обиколки в секунда между алгоритъм с изчерпателно търсене и алгоритъма на Bellman-Held-Karp

Интересното е, че за 60 години откакто е предложен този по-бърз алгоритъм, не е постигнато значително подобрение във времето за намиране на точно решение. И въпреки че няма доказателство, че не съществува алгоритъм, който решава проблема в полиномно време, много учени и изследователи смятат, че такъв не съществува. От това следва, че алгоритъмът на Хелд-Карп е може би най-подходящият кандидат за решение на нашата задача. Нека разгледаме самия алгоритъм по-подробно.

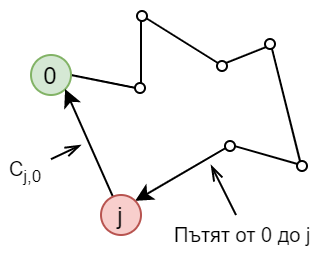
## 5.1 Алгоритъм

В основата на алгоритъма на Белман-Хелд-Карп лежи парадигмата на динамичното програмиране, която накратко можем да формулираме така:

1. Идентифицираме сравнително малка колекция от под-проблеми
2. Показваме как бързо и коректно да решим „големи“ под-проблеми, като преизползваме решенията за „малките“ под-проблеми
3. Показваме как бързо и коректно да направим заключение за финалния резултат на базата на решенията на всички под-проблеми.

Ключът към прилагането на динамичното програмиране е идентифицирането на правилната колекция от под-проблеми. Най-добрият начин да открием такива под-проблеми е да анализираме как оптималното решение може да е съставено от оптимални решения на по-малки под-проблеми.

Да предположим, че разполагаме с оптимална обиколка , която преминава през върховете от , където Как би изглеждала тази обиколка? По-колко начина би могла да бъде съставена от оптимални решения на по-малки под-проблеми. Нека нашата обиколка започва и завършва на 0. Ако разгледаме последното ребро, което се връща обратно в 0 от някакъв връх j, то тогава нашата обиколка ще бъде съставена от минималния път, без цикли от 0 до j последван от последното ребро от j обратно до 0. След като j може да бъде всеки един връх от 1 до n, то в този случай съществуват точно n-1 на брой пътя с j за последен връх. Например ако означим с път, който започва от 0, преминава през всички върхове на S и завършва на j, където , то в нашия конкретен пример можем да запишем min( което ще означава, че тръгваме от 0, минаваме по-някакъв път през 1,2 и 3 и завършваме в 4 и това е минималния път от 0 до 4.



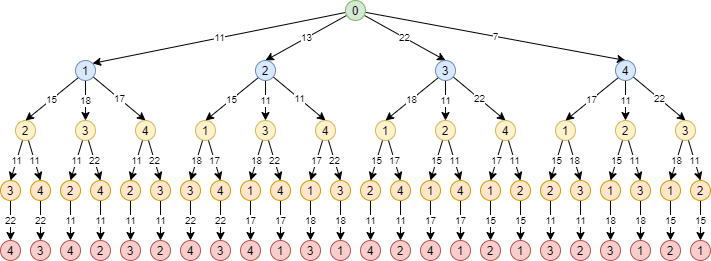
Фигура 5   
Пътят от 0 до j минава през всички върхове на Т и е с минимална дължина.

Виждаме че за нашата задача можем да имаме точно 4 такива пътя и единият от тях ще е оптималният. Нека с записваме директното разстояние между 2 върха k и j. Тогава формулата за нашата оптимална обиколка ще изглежда така:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1) | *min* |  | *min* |  |
|  | *min* |  |
|  |
|  | *min* |  |
|  |
|  |  |  | *min* |  |
|  |
|  | *min* |  |
|  |
|  | *min* |  |
|  |

Нека разгледаме дърво, което изброява всички възможни пътища в примерния тест от условието на задачата (Фигура 5) . На фигурата виждаме, че за всеки възможен път липсва само последната отсечка, която се връща обратно в началната бензиностанция (0).

Фигура 6  
Дърво представляващо всички възможни пътища от примерната конфигурация на бензиностанциите от условието на задачата. За всеки път липсва единствено последната отсечка, която затваря обиколката.



Алгоритъма се състой от следните стъпки:



Нека M e масив