Дніпропетровський ліцей інформаційних технологій при  
Дніпропетровському національному університеті

Кафедра Інформатики

**КУРСОВА РОБОТА**

**Тема:** Електронний довідник «Задачі динамічного програмування за профілем»

**Виконав:**

Ліцеїст 9-В класу

Мордак Антон Олександрович

**Керівники:**

Ентін Йосиф Абрамович,

**Робота допущена до захисту**

\_\_\_\_\_ Ентін Йосиф Абрамович

Дніпропетровськ

2020

Електронний довідник «Задачі динамічного програмування за профілем»

Виконав Мордак Антон Олександрович

Учень 10 класу Ліцею інформаційних технологій

При Дніпропетровському національному університеті

Імені Олеся Гончара

Місто Дніпропетровськ

Науковий керівник:

Ентін Йосиф Абрамович

**Мета роботи**: Створити електронний довідник де розглянути основні задачі та методи рішень динамічного програмування за профілем. Замощення довільної площини часто використовуються на практиці, одним з яскравих прикладів є кладка паркету, потрібно вирахувати кількість варіантів замощення площини, використання динамічного програмування за профілем стало невід’ємною частиною життя для багатьох компаній, оскільки завдяки цьому складність алгоритму багатьох програм значно зменшилася, також задачі такого типу часто використовують на співбесідах перед прийняттям на роботу нового працівника. Тому ця робота є актуальною. Метою було створити електронний довідник розгляду динамічного програмування за профілем розглянемо такі задачі:

1. Розглянути динамічне програмування за профілем.
2. Розрахунок складності алгоритму динаміки по профілю
3. Створити електронний довідник.

**ЗМІСТ**

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

|  |  |
| --- | --- |
| ДПП | Динамічне програмування за профілем. |
| ДП | Динамічне програмування. |
| Псевдокод | Неформальний запис алгоритму, який використовує структуру поширених мов програмування, але нехтує деталями коду, неістотними для розуміння алгоритму (опис типів, виклик підпрограм тощо). |
| Профіль | Один із стовпців (рядків), що задовольняє умові завдання. |
| Правила домінування | Такі правилами які дозволяють робити порівняння варіантів розвитку послідовностей і завчасне відсіювання безперспективних варіантів. |
| Розв’язувальні правила | Такі правила домінування, що вони визначають елементи оптимальної послідовності однозначно один за одним. |

**ВСТУП**

На практиці часто доводиться зустрічатись з такими ситуаціями, коли метою оптимізації є встановлення найкращої послідовності тих чи інших дій (виробничих операцій, етапів будівництва споруд, виконання курсової роботи тощо). З подібною метою зустрічаються й програмісти при розв’язанні задач динамічного програмування чи на співбесіді.

Динамічне програмування – це спосіб вирішення складних задач шляхом розбиття їх на більш прості підзадачі. Це застосовується до завдань з підструктурою, що виглядає як набір декількох задач, складність яких менше вихідної. В цьому випадку час обчислень можна значно скоротити.

Основний спосіб полягає в знаходженні таких правил, які дозволяють робити порівняння варіантів розвитку послідовностей і завчасне відсіювання безперспективних варіантів. У ряді випадків в задачах динамічного програмування вдається одержати такі сильні правила домінування, що вони визначають елементи оптимальної послідовності однозначно один за одним. В такому випадку правила домінування називають розв’язувальними правилами.

В цій курсовій роботі ми розглянемо один з багатьох типів задач динамічного програмування - динамічне програмування за профілем.

Типовими завданнями, на динаміку за профілем, є задачі на замощення деякої площини фігурами, але питатися можуть різні речі, наприклад:

* знайти кількість способів замощення поля деякими фігурами
* знайти замощення з найменшою кількістю фігур
* знайти замощення з мінімальною кількістю невикористаних клітин
* знайти замощення з мінімальною кількістю фігур таке, що в нього не можна додати ще одну фігуру

Метою роботи було розроблення електронного довідника в якому було розібране ДПП на прикладі задач, та розрахунок переваги такого метода рішення над повним перебором.

Об’єктом цього дослідження був двовимірний масив, представлений у вигляді матриці заповненої або 1 або 0 розмірами N\*M , де N- довжина заданої площини по горизонталі, і вводиться користувачем, M- довжина заданої площини по вертикалі, також вводиться користувачем. Визначалася кількість варіантів замощення цієї площини доміно. Методами дослідження масиву було ДПП за допомогою мови програмування С++.

Даний проект самостійно розроблений автором без сторонньої допомоги. При розробці проекту використовувалася тільки література з даної теми. Готові програми з теми не використовувалися.

Виконані дослідження можуть бути корисними для розробників нових композиційних матеріалів.

**РОЗДІЛ 1 Загальні відомості**

В основі динамічного програмування стоїть принцип оптимальності сформульований Р. Беллманом: «Відрізок оптимального процесу від будь-якої його точки до кінця процесу сам є оптимальним процесом з початком в даній точці».

Динамічне програмування - метод оптимізації, пристосований до операцій, в яких процес прийняття рішень може бути розділений на окремі етапи (кроки).

Сам термін динамічне програмування був запропонований в 1940-х роках Річардом Беллманом для характеристики процесу розв'язування проблем, при якому потрібно знаходити найкращі рішення, одне за одним.

Динамічне програмування за профілем – підрозділ дп, спосіб оптимізації перебора кількості варіантів за допомогою методів дп, в тому випадку коли один з вимірів невеликий. Майже всі задачі зводяться до замощення деякої площини, деякими фігурами.

**Розділ 2 Розрахування складності алгоритму**

Безперечно усі ці задачі можна вирішувати повним перебором. Навіщо тоді потрібна динаміка? Порахуємо складність алгоритму для динаміки та повного перебору. Для повного перебору складність рахується за формулою О(аNM)

а – Кількість різноманітних варіантів замощення однієї клітини.

N – Довжина замощуваної площини.

M – Ширина замощуваної площини.

Розглянемо типову задачу «Доміно», задача звучить так: Заповнити площину n на m доміношками розмірами 2x1 1x2. В кожної клітинки на площині n, m є 2 варіанта замощення. Тому вже для значення 10 в варіанті повного перебору буде 2100, що є дуже велика величина.

Тепер розрахуємо складність алгоритму для динаміки.

Також розрахуємо кількість варіантів замощення однієї доміношки очевидно вона розраховується за формулою a = mas\*M

а – Кількість різноманітних варіантів замощення однієї клітини.

M – Ширина замощуваної площини.

mas – В нашому випадку буде рівна 22n

Отже вихідна складність алгоритму буде О(22n\*m)

Неозброєним оком видно, що складність алгоритму менша в багато разів. Отже використання динаміки доцільно. Але оскільки в степені є n, то певні обмеження ми маємо. При великому значенні n, в нас виникають проблеми з часом.

Якщо значення m невелике, то все, що нам потрібно зробити це поміняти місцями n та m. Та знова отримати малу складність алгоритму, це буде неможливо при великому значенні m. Отже в нас виникають обмеження на n та m.

І нам потрібно встановити максимальні значення розміру площини при якому алгоритм буде ефективний. Для цього створемо таблицю складності алгоритму для значень n та m з діапазоном від 1 до 10.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n\m | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 4 | 16 | 64 | 256 | 1,024 | 4,096 | 16,384 | 65,536 | 262,144 | 1,048,576 |
| 2 | 8 | 32 | 128 | 512 | 2,048 | 8,192 | 32,768 | 131,072 | 524,288 | 2,097,152 |
| 3 | 12 | 48 | 192 | 768 | 3,072 | 12,288 | 49,152 | 196,608 | 786,432 | 3,145,728 |
| 4 | 16 | 64 | 256 | 1,024 | 4,096 | 16,384 | 65,536 | 262,144 | 1,048,576 | 4,194,304 |
| 5 | 20 | 80 | 320 | 1,280 | 5,120 | 20,480 | 81,920 | 327,680 | 1,310,720 | 5,242,880 |
| 6 | 24 | 96 | 384 | 1,536 | 6,144 | 24,576 | 98,304 | 393,216 | 1,572,864 | 6,291,456 |
| 7 | 28 | 112 | 448 | 1,792 | 7,168 | 28,672 | 114,688 | 458,752 | 1,835,008 | 7,340,032 |
| 8 | 32 | 128 | 512 | 2,048 | 8,192 | 32,768 | 131,072 | 524,288 | 2,097,152 | 8,388,608 |
| 9 | 36 | 144 | 576 | 2,304 | 9,216 | 36,864 | 147,456 | 589,824 | 2,359,296 | 9,437,184 |
| 10 | 40 | 160 | 640 | 2,560 | 10,240 | 40,960 | 163,840 | 655,360 | 2,621,440 | 10,485,760 |

З таблиці видно, що вже при значенні довжини та площини рівним 10 алгоритм починає погано працювати. Отже тепер будемо відштовхуватись від того, що n<10 та m<10.

За цією таблицею створемо графік функції. Він має такий вигляд

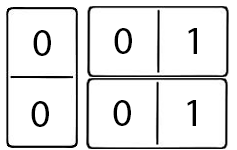
**Розділ 3.1 Теоретичне вирішення задачі**

Знайти кількість способів замостити таблицю n × m за допомогою доміношек розмірами 1 × 2 або 2 × 1. За умови, що n та m < 10. Причому:

1. Площина повинна бути повністю заповнена.
2. Доміношки не повинні накладатися
3. Доміношки не повинні вилазити за краї площини

Розглянемо метод динамічного програмування за профілем на прикладі цієї задачі.

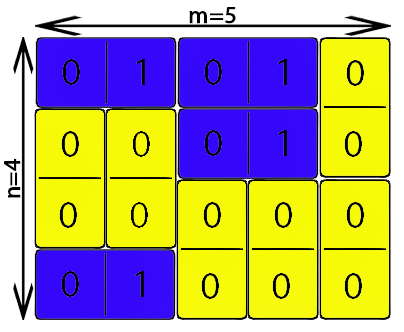
Рахувати кількість способів будемо поступово. Спочатку ми розглядаємо перший стовпчик і тільки після того як заповнили його розглядаємо наступний і т.д.

Звіряти чи є такий спосіб будемо за профілями, якщо з одного профіля можна перейти в наступний, то даний спосіб може існувати. А перехід між профілями перевіряється за такими правилами:

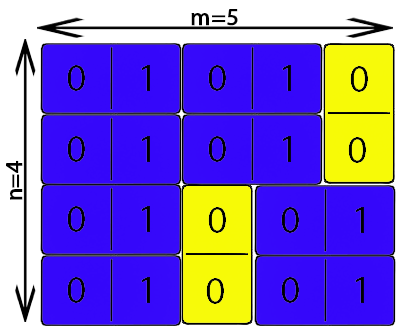
Нехай, якщо доміношка знаходиться у вертикальному положенні позначимо її частини як 0 0. Якщо у горизонтальному то правий кінець доміношки позначимо за 1, а лівий край 0. Див. малюнок 1. Це зроблене для того, щоб ми не заповнювали одну клітинку двічі.

Малюнок

Тоді має право твердження, що біля 1 у рядку не може стояти 1. А пари із 0 в рядку повинні мати аналогічні пари в рядках зверху або знизу. Зобразимо площину у вигляді матриці заповненої 0 та 1. У першому стовпці матриці завжди будуть 0 оскільки доміношки не повинні вилізати за межі площини з цього і починаємо. Біля 0 може стояти як 0 так і 1, а біля 1 тільки 0. Також, щоб перейти до нового профілю доміношки повинні покривати лише незайняті клітинки, а щоб перейти на новий профіль, старий повинен бути повністю заповненим. Розглянемо один з варіантів замощення площини n\*m, де n = 4, m=5 Див. малюнок 2. На малюнку зображена матриця заповнення клітинок доміно.



В цьому варіанті, нульовий профіль заповнений двома горизонтальними та однією вертикальною.

Перший профіль вже наполовину заповнений, другою частиною горизонтальних доміношек. Тому заповнені клітинки ми не раховуємо, починаємо з клітинки з координатами x = 1,y = 1 по матриці. 

Розгянемо інший варіант заповнення Див. малюнок 3. Щоб не рахувати одні і ті ж випадки перевіремо:

Якщо в стовпці немає вертикальних доміношек(тобто лише горизонтальні) то рахувати наступний профіль не має сенсу, оскільки там будуть тільки 1, а оскільки біля 1 може стояти тільки 0 то і наступний буде складатися лише з 0. Це яскраво видно по профілям з координатами 0 та 1.

**Розділ 3.2 Практичне вирішення задачі**

public int n, m;

long[,] mas;

private void button1\_Click(object sender, EventArgs e)

{

n = int.Parse(textBox1.Text);

m = int.Parse(textBox3.Text);

mas = new long[n + 1, 1 << m];

mas[0, 0] = 1;

for (int x = 0; x < n; x++)

{

for (int mask = 0; mask < (1 << m); mask++)

{

calc(x, 0, mask, 0);

}

}

label5.Text = Convert.ToString(mas[n, 0]);

}

public void calc(int x = 0, int y = 0, int mask = 0, int next\_mask = 0)

{

if (x == n)

{

return;

}

if (y >= m)

{

mas[x + 1, next\_mask] += mas[x, mask];

}

else

{

int my\_mask = 1 << y;

if ((mask & my\_mask) != 0)

{

calc(x, y + 1, mask, next\_mask);

}

else

{

calc(x, y + 1, mask, next\_mask | my\_mask);

if (y + 1 < m && (mask & my\_mask) == 0 && (mask & (my\_mask << 1)) == 0)

{

calc(x, y + 2, mask, next\_mask);

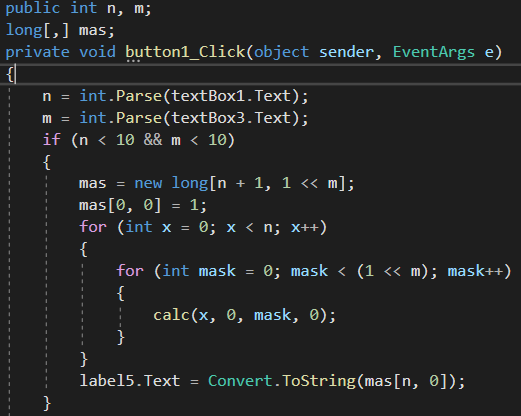
}

}

}

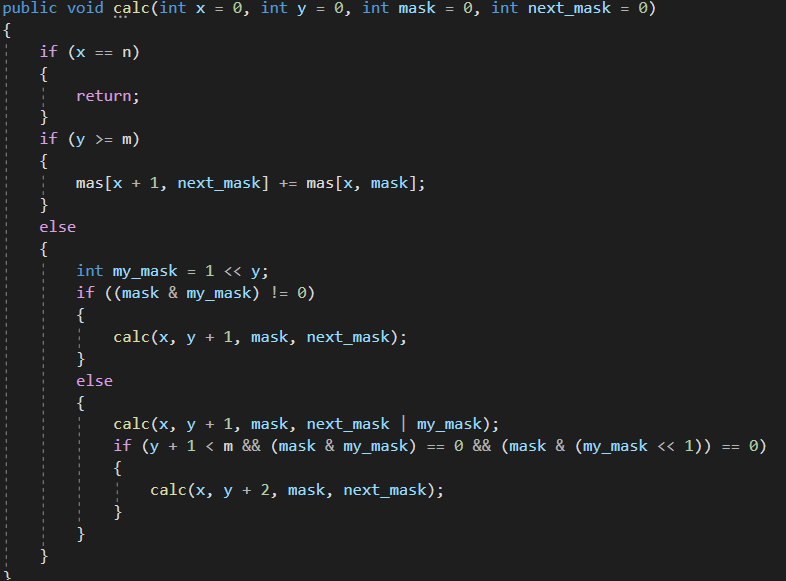
}

Спочатку хочу розглянути основний код задачі.



В цій задачі доцільно використовувати звичайний двовимірний масив оскільки розміри площини зразу задані та по ходу процесу змінюватися не будуть. Але функція перевірки можливості переходу до іншого профілю також використовує масив, і щоб не передавати до функції посилання на масив, я його обьявив глобально. Спочатку ми вводимо значення довжини та ширини площини яку нам слід замостити та перевіряємо чи відповідають умові задачі. Вводимо розміри масиву на базі якого ми будемо виконувати наші обчислення. Потім за допомогою подвійного циклу ми викликаємо процедуру перевірки можливості переходу. Зверніть увагу клітинці за координатами (0;0) ми зразу присвоюємо 1, оскільки з цієї клітини 100% можна перейти до наступного профіля.

Очевидно, що кількість варіантів буде знаходитись у координатах mas[n,0].

Тепер розберемо основну функцію перевірки можливості переходу.

**Розділ 5 Технічні вимоги для поекту**

Для користування проектом на комп’ютері повинна бути встановлена операційна система Windows 7 або пізніші версії. Також повинна бути встановлена Visual Studio 2019 або пізніші версії. На жорсткому дисці повинно бути вільні принаймні 5MB, та від 16 MB вільної оперативної пам’яті. Частота процессора повинна бути не менша від 2 GHz.

**Розділ 4 Висновки**

Виконуючи курсову роботу я навчився використовувати динамічне програмування для вирішення складних задач та працювати з двовимірними масивами, розраховувати складність алгоритму. За результатами роботи було створено працюючий електронний довідник, який може використовуватися як научна база для інших дослідницьких робіт. Також була створена програма вирішення задачі «Доміно» на базі якої і була розглянута тема. Я збираюся продовжити цю роботу у своїй випускній роботі та розглянути інший спосіб вирішення таких задач, а саме за допомогою зламаного профіля. Або розглянути цю ж задачу, але у тривімірному просторі. Або замощувати площину довільними фігурами. Також існує варіант продовження цієї ж теми, але оптимізувати алгоритм для більших значень довжини та ширини.

**Розділ 5 Cписок електронних джерел інформації**

<https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BF%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8E>

<https://wiki.algocode.ru/index.php?title=%D0%94%D0%9F_%D0%BF%D0%BE_%D0%BF%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8E>

<https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5>

<http://www.pereplet.ru/nauka/Soros/pdf/9802_139.pdf>

<https://habr.com/ru/post/191498/>

<https://e-maxx.ru/algo/profile_dynamics>

<https://ru.coursera.org/lecture/sportivnoe-programmirovanie/4-5-zadacha-parkiet-dinamika-po-profiliu-k3EYK>

<https://archive.lksh.ru/2011/july/B/files/dp-profile.pdf>