**Разбиване на число**

**Дадено е естествено число n. Под разбиване на числото n ще разбираме всяко множество от естествени числа, чиято сума е n. Например всички възможни разбивания за 4 са: 4, 3+1, 2+2, +1+1+2, 1+1+1+1.**

**Задачата е да се намери броя на всички разбивания на естественото число n.**

**І начин: пълно изчерпване** – ще генерираме всички разбивания на числото и когато дадено разбиване е готово, го преброяваме. Започваме от събираемо 1 и за да избегнем разбивания като 5=2+3 и 5=3+2 ще пазим стойността на предходното събираемо – променлива prev, и ще проверяваме дали текущото е >= на предходното.

#include <iostream>

using namespace std;

int br=0;

void razbivane(int n, int prev)

{

if (n==0)br++;

else

for (int i=1;i<=n;i++)

if (i>=prev)

razbivane(n-i, i);

}

int main()

{

int n;

cin>>n;

razbivane(n,0);

cout<<br<<endl;

return 0;

}

**ІІ начин – динамирно оптимиране**

Разглеждаме по-общата задача – да определим броя S(n, m) на всички разбивания на числото n на сума от числа, ненадминаващи m. Когато решим тази задача, търсеният брой разбивания на n ще бъде S(n,n).

Разписваме разбиванията на първите 6 числа на събираеми, ненадминаващи определена стойност m, за да открием зависимост.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Число** | **Сума от събираеми, ненадминаващи m=** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 |
| **2** | 2 | 2  1+1 | 2  1+1 | 2  1+1 | 2  1+1 | 2  1+1 |
| **3** | 3 | 3  1+2 | 3  1+2  1+1+1 | 3  1+2  1+1+1 | 3  1+2  1+1+1 | 3  1+2  1+1+1 |
| **4** | 4 | 4  1+3  2+2 | 4  1+3  2+2  1+1+2 | 4  1+3  2+2  1+1+2  1+1+1+1 | 4  1+3  2+2  1+1+2  1+1+1+1 | 4  1+3  2+2  1+1+2  1+1+1+1 |
| **5** | 5 | 5  1+4  2+3 | 5  1+4  2+3  1+1+3  1+2+2 | 5  1+4  2+3  1+1+3  1+2+2  1+1+1+2 | 5  1+4  2+3  1+1+3  1+2+2  1+1+1+2  1+1+1+1+1 | 5  1+4  2+3  1+1+3  1+2+2  1+1+1+2  1+1+1+1+1 |
| **6** | 6 | 6  1+5  2+4  3+3 | 6  1+5  2+4  3+3  1+1+4  1+2+3  2+2+2 | 6  1+5  2+4  3+3  1+1+4  1+2+3  2+2+2  1+1+1+3  1+1+2+2 | 6  1+5  2+4  3+3  1+1+4  1+2+3  2+2+2  1+1+1+3  1+1+2+2  1+1+1+1+2 | 6  1+5  2+4  3+3  1+1+4  1+2+3  2+2+2  1+1+1+3  1+1+2+2  1+1+1+1+2  1+1+1+1+1+1 |

В следващата таблица отразяваме броя на разбиванията:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Число** | **Брой разбивания** | | | | | |
| **от 1 число** | **от 2 числа** | **от 3 числа** | **от 4 числа** | **от 5 числа** | **от 6 числа** |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 2 | **2** | 2 | 2 | 2 |
| **3** | 1 | 2 | 3 | 3 | 3 | 3 |
| **4** | 1 | 3 | 4 | 5 | 5 | 5 |
| **5** | 1 | **3** | **5** | 6 | 7 | 7 |
| **6** | 1 | 4 | 7 | 9 | 10 | 11 |

От втората таблица се вижда, че първи ред и първа колона имат стойност 1.

Числата вдясно от главния диагонал са равни на числата по главния диагонал.

Числата по главния диагонал са с 1 по-големи от предходните им.

Числата под главния диагонал са сума от стойностите на две клетки – тази вляво от текущата, и тази същата колона, но на ред (n-m) , т.е  
а[5][3] = a[5][2]+a[2][3].

Обобщаваме получените резултати в следната формула:

S(n, m) =

Задачата се свежда до попълване на масив a[n][n] по зададената формула и извеждането на елемента a[n][n].

Решение:

#include <iostream>

using namespace std;

int n, a[1001][1001];

void calculate()

{

for (int i=1; i<=n; i++) a[i][1]=a[1][i]=1;

for (int i=2; i<=n; i++)

for (int j=2; j<=n; j++)

if (i==j) a[i][j]=1+a[i][j-1];

else if (i<j) a[i][j]=a[i][i];

else a[i][j]=a[i][j-1]+a[i-j][j];

}

int main()

{

cin >> n;

calculate();

cout << a[n][n] << endl;

return 0;

}

Решение на по-общата задача – да се намери броя на всички разбивания на числото n на сума от числа, ненадминаващи m.

#include <iostream>

using namespace std;

int n, m, a[1001][1001];

void calculate()

{

for (int i=1; i<=n; i++) a[i][1]=a[1][i]=1;

for (int i=2; i<=n; i++)

for (int j=2; j<=m; j++)

if (i==j) a[i][j]=1+a[i][j-1];

else if (i<j) a[i][j]=a[i][i];

else a[i][j]=a[i][j-1]+a[i-j][j];

}

int main()

{

cin >> n >> m;

calculate();

cout << a[n][m] << endl;

return 0;

}

**Разширена задача: Задачата може да се формулира и така: Да се изведат всички разбивания на естественото число n.**

Тогава решението е чрез пълно изчерпване на всички възможности. Ще съхраняваме разбиванията в масив. Ще започнем разбиването с максималното събираемо – например 5 = 5, после 5=4+1, след това 5 = 3+2 и т.н . За да избегнем разбивания от вида 5=2+3 и 5=3+2 ще проверяваме дали текущата стойност е <= от предходната. Затова се налага да дадем стойност на първи елемент да е равен на n или друго по-голямо число. Функцията razbivane е рекурсивна, тя получава като входни параметри числото n и pos – текущата позиция в масива.

#include <iostream>

using namespace std;

int a[1000];

void print(int br)

{

for (int i=1;i<=br;i++)

cout<<a[i]<<" ";

cout<<endl;

}

void razbivane(int n, int pos)

{

if (n==0)print (pos-1);

else

for (int i=n;i>=1;i--)

if (i<=a[pos-1])

{

a[pos]=i;

razbivane(n-i, pos+1);

}

}

int main()

{

int n;

cin>>n;

a[0]=n;

razbivane(n,1);

return 0;

}

**Генериране на разбивания на число като сума с рекурсия.**

#include <iostream>

using namespace std;

int l = 1;

int m[100] = {1};

int n;

void razb()

{ const int n2 = n / 2;

int& ml = m[l];

for (ml = m[l - 1]; ml <= n2; ml++)

{ n -= ml;

l++;

razb();

l--;

n += ml;

}

for (int i = 1; i < l; i++)

cout << m[i] << " ";

cout << n << endl;

}

int main()

{ cin >> n;

razb();

return 0;

}