|  |
| --- |
| Комбинаторный анализ – это раздел математики, посвященный решению задач выбора и расположения элементов некоторого, множества в соответствии с заданными правилами. |
|  |
| Какую задачу нецелесообразно решать при помощи рекурсивных алгоритмов: |
| ? |

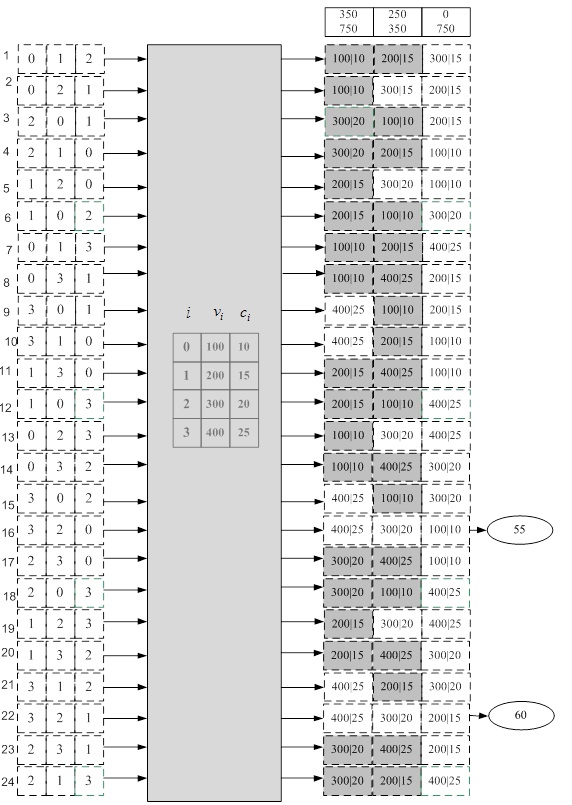
Ниже представлен фрагмент программного кода, реализующую функцию knapsack\_s. Для решения какой задачи используется эта функция?  
  
// Knapsack.cpp  
#include "stdafx.h"  
#include "Knapsack.h"  
#define NINF 0x80000000   
int calcv(combi::subset s, const int v[])   
{  
int rc = 0;   
for (int i = 0; i < s.sn; i++) rc += v[s.ntx(i)];   
return rc;  
};  
int calcc(combi::subset s, const int v[], const int c[])   
{  
int rc = 0;   
for (int i = 0; i < s.sn; i++) rc += (v[s.ntx(i)]\*c[s.ntx(i)]);   
return rc;  
};  
void setm(combi::subset s, short m[])  
{  
for (int i = 0; i <s.n; i++) m[i] = 0;  
for (int i = 0; i < s.sn; i++) m[s.ntx(i)] = 1;  
};  
int knapsack\_s(  
int V,   
short n,   
const int v[],   
const int c[],   
short m[]   
)  
{  
combi::subset s(n);  
int maxc = NINF, cc = 0;  
short ns = s.getfirst();   
while (ns >= 0)  
{  
if (calcv(s, v) <= V)  
if ((cc = calcc(s,v,c)) > maxc)   
{  
maxc = cc;  
setm(s,m);  
}  
ns = s.getnext();   
};  
return maxc;   
};

упрощенной задачи о рюкзаке;

Решение упрощенной задачи о рюкзаке основано на:

генераторе множества всех подмножеств;

На рисунке изображена схема, поясняющая решение о загрузке судна с условием центровки. Задача имеет следующие исходные данные: n = 4 – общее количество контейнеров;m = 3 – количество свободных мест на палубе судна; (100,200,300,400) – вес контейнеров; (10,15,20,25) – доход от перевозки контейнеров; (350,250,0) – минимальный вес контейнеров;(750,350,750) – максимальный вес контейнеров. В какой строке представлен оптимальный план размещения контейнеров на палубе



22

Из скольких этапов состоит решение транспортной задачи:

2

Алгоритм Джонсона-Троттера применяется для получения

Множества всех перестановок

Алгоритм поиска в ширину заключается в:

посещении вершин в порядке их удаленности от некоторой заранее выбранной или указанной стартовой вершины;

Какую задачу нецелесообразно решать при помощи динамического программирования:

?

Одним из определений динамического программирования является:

Метод решения задачи оптимизации, реализующей рекурсивный алгоритм с перекрывающимися подзадачами, в котором каждая такая подзадача решается один раз, а ее результат сохраняется для последующего применения.

Решение задачи о коммивояжере основано на:

генераторе множества всех перестановок;

Математическое программирование – это

область математики, изучающая теорию и численные методы решения многомерных экстремальных задач

Имеется некая задача линейного программирования с двумя неизвестными. Какими методами следует решать данную задачу:

графическим и симплекс-методом;

Фрагмент программного кода, представленный ниже, используется для программной реализации  
// Combi.cpp   
#include "stdafx.h"  
#include "Combi.h"  
#include <algorithm>  
namespace combi  
{  
xcombination::xcombination (short n, short m)  
{  
this->n = n;  
this->m = m;  
this->sset = new short[m+2];  
this->reset();   
}  
void xcombination::reset() // сброситьгенератор, начатьсначала  
{  
this->nc = 0;  
for(int i = 0; i < this->m; i++) this->sset[i] = i;   
this->sset[m] = this->n;  
this->sset[m+1] = 0;   
};   
short xcombination::getfirst()  
{ return (this->n >= this->m)?this->m:-1; };  
short xcombination::getnext() // сформировать следующий массив индексов  
{  
short rc = getfirst();   
if (rc > 0)  
{  
  
short j;  
for (j = 0; this->sset[j]+1 == this->sset[j+1]; ++j)   
this->sset[j] = j;  
if (j >= this->m) rc = -1;  
else {  
this->sset[j]++;  
this->nc++;  
  
};  
  
}  
return rc;   
};   
short xcombination::ntx(short i)  
{return this->sset[i]; };  
  
unsigned \_\_int64 fact(unsigned \_\_int64 x){return(x == 0)?1:(x\*fact(x-1));};  
  
unsigned \_\_int64 xcombination::count() const  
{  
return (this->n >= this->m)?  
fact(this->n)/(fact(this->n-this->m)\*fact(this->m)):0;   
};  
};

генератора множества всех сочетаний

Решение задачи о загрузке судна основано на:

генераторе множества всех сочетаний;

Имеется множество, состоящее из n элементов. Количество элементов множества всех подмножеств вычисляется по формуле

2^n

Условие задачи о загрузке судна выглядит следующим образом:   
На палубе судна имеется m мест для размещения стандартных контейнеров. Выбрать n контейнеров для погрузки на судно можно из n больше m имеющихся в наличии. Каждый контейнер i характеризуется весом v и доходом c от его перевозки. Необходимо выбрать m контейнеров таким образом, чтобы их общий вес не превышал V, но при этом доход от перевозки был максимально возможным. Как может быть записана математическая модель задачи (ki – неизвестные (номера выбранных контейнеров), которые требуется найти):



Ниже представлен фрагмент программного кода, обеспечивающий вычисление продолжительности решения задачи о рюкзаке при различном количестве предметов. Какая стандартная функция отвечает за вычисление продолжительности решения задачи?   
clock();

Имеется множество, состоящее из nэлементов. Количество перестановок данного множества рассчитывается по формуле

n!

Ниже представлен фрагмент программного кода функции boat\_c, применяемой при решении задачи о загрузке судна с условием центровки. Укажите какое из утверждений является неправильным:  
// --- Вoat.cpp  
#include "stdafx.h"  
#include "Boat.h"  
#include "Combi.h"  
namespace boatfnc  
{  
bool compv( combi::accomodation s, const int ming[],   
const int maxg[], const int v[])  
{  
int i = 0;  
while(i < s.m && v[s.ntx(i)] <= maxg[i] && v[s.ntx(i)] >= ming[i])i++;   
return (i == s.m);  
};  
int calcc(combi::accomodation s,const int c[])   
{  
int rc = 0;   
for (int i = 0; i < s.m; i++) rc += c[s.ntx(i)];   
return rc;  
};  
voidcopycomb(short m, short \*r1, const short \*r2)  
{ for (int i = 0; i < m; i++) r1[i] = r2[i]; };  
}  
intboat\_с( // функция возвращает доход от перевозки контейнеров  
short m,// [in] количество мест для контейнеров  
intminv[], // [in] минимальный вес контейнера на каждом месте  
intmaxv[], // [in] максимальный вес коннтейнера каждом месте  
short n,// [in] всего контейнеров  
const int v[],// [in] вес каждого контейнера  
const int c[],// [in] доход от перевозки каждого контейнера  
short r[]// [out] номера выбранных контейнеров  
)  
{  
combi::accomodation s(n, m);  
int rc = 0, i = s.getfirst(), cc = 0;   
while (i > 0)  
{  
  
if (boatfnc::compv(s, minv, maxv, v))   
  
  
if ((cc = boatfnc::calcc(s,c)) > rc)   
{rc = cc; boatfnc::copycomb(m, r, s.sset);}  
  
i = s.getnext();   
};  
return rc;  
};

В функции boat\_cприменяется генератор размещений (combi::accomodation). Конструктору генератора передается один параметр: n(количество контейнеров).

Функция boat\_c вызывает вспомогательную функцию boatfnc::compv, которая позволяет проверить допустимость размещения контейнеров. Если вес всех контейнеров в размещении удовлетворяет заданным ограничениям, то функция возвращает true, иначе возвращается false.