**Практическая работа № 16,17**

**«**Решение задач динамического программирования **»**

**Цель работы:** Научиться решать задачи динамического программирования.

**Образовательные результаты, заявленные во ФГОС третьего поколения:**

Студент должен

уметь:

- подбирать аналитические методы исследования математических моделей;

- использовать численные методы исследования математических моделей;

знать:

- методы исследования математических моделей разных типов.

**Краткие теоретические и учебно-методические материалы по теме практической работы**

**1. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

Динамическое программирование представляет собой математический метод, заслуга создания и развития которого принадлежит, прежде всего, Беллману. Метод можно использовать для решения весьма широкого круга задач, включая задачи распределения ресурсов, замены и управления запасами, задачи о загрузке. Характерным для динамического программирования является подход к решению задачи по этапам, с каждым из которых ассоциирована одна управляемая переменная. Набор рекуррентных вычислительных процедур, связывающих различные этапы, обеспечивает получение допустимого оптимального решения задачи в целом при достижении последнего этапа. Несомненна привлекательность идеи разбиения задачи большой размерности на подзадачи меньшей размерности, включающие всего по нескольких переменных, и последующего решения общей задачи по частям. Именно на этой идее основан метод динамического программирования.

Происхождение названия динамическое программирование, вероятно, связано с использованием методов динамического программирования в задачах принятия решений через фиксированные промежутки времени (например, в задачах управления запасами). Однако методы методов динамического программирования успешно применяются также для решения задач, в которых фактор времени не учитывается. По этой причине более удачным представляется термин многоэтапное программирование, отражающий пошаговый характер процесса решения задачи.

Фундаментальным принципом, положенным в основу теории методов динамического программирования, является принцип оптимальности. По существу, он определяет порядок поэтапного решения допускающей декомпозицию задачи (это более приемлемый путь, чем непосредственное решение задачи в исходной постановке) с помощью рекуррентных вычислительных процедур.

**1.1. Постановка задачи динамического программирования**

Рассмотрим функционирование некоторого объекта в течение промежутка времени *Т*. Пусть в момент времени *tj* состояние объекта характеризуется вектором , причем в начальный момент времени *t1* состояние объекта задано, т.е. вектор  известен. Объект меняет свое состояние под воздействием управления  в моменты времени  . При этом управляющее воздействие выбирается из заданной области , т.е.:

 (1.1)

Рассматривается случай, когда будущее состояние объекта зависит только от состояния объекта в данный момент времени и управляющего воздействия в этот момент времени, т.е.

 . (1.2)

Здесь  – заданная функция своих аргументов. Уравнение (3.2) является уравнением движения объекта, т.е. описывает динамику объекта во времени.

Обозначим  – выигрыш, получаемый от функционирования объекта на *j*-м участке времени, т.е. в полуинтервале . Если в момент времени , когда объект находился в состоянии  выбрать управление , то объект перейдет в состояние . Далее последовательно можно выбирать в соответствии с (3.1) , в из (3.2) определять . Этим значениям  и  будет соответствовать вполне определенное значение дохода:

 (1.3)

Если выбирать другие значения управляющих воздействий, то им будут соответствовать другие состояния объекта, а следовательно, и другое значение общего дохода (3.3). Поэтому естественно поставить следующую оптимизационную задачу:

 (1.4)

 . (1.5)

 (1.6)

- задан. (1.7)

Здесь необходимо найти неизвестные  и , которые удовлетворяли бы ограничениям (1.5)-(1.7) и максимизировали бы целевую функцию (1.4).

Эта задача является в общем случае задачей нелинейного программирования и обладает следующими особенностями:

1. Искомые переменные разбиты на *N* групп, в каждую *j*-тую из которых входят только  и .
2. Целевая функция (1.4) является суммой функций , каждая из которых зависит лишь от переменных соответствующей группы. В этом случае говорят, что целевая функция сепарабельна, или аддитивна.
3. Уравнение (1.5) является рекуррентным, т.е. через значения  и  однозначно определяется .

Многие реальные задачи исследования операций сводятся к математическим моделям вида (3.4)-(3.7), которая допускает различные интерпретации.

Решение задачи вида (3.4)-(3.7) обычными методами оказывается либо невозможным, либо неэффективным из-за большой размерности. Поэтому ее решение сводится к последовательному решению *N* связанных между собой задач меньшей размерности. Для выявления этих задач и связей между ними рассмотрим задачу вида (1.4)-(1.7), соответствующую последним этапам . Она запишется в виде:

 (1.8)

 . (1.9)

 (1.10)

- фиксирован. (1.11)

В этой задаче мы не знаем, чему равно конкретное значение , но если его зафиксировать, то получим соответствующее максимальное значение (3.8). Если зафиксировать другое значение , то естественно максимальное значение целевой функции (3.8) будет другим. Обозначим максимальное значение целевой функции (3.8) при некотором зафиксированном  через :



Запишем это равенство в виде:

 (1.12)

Здесь первое слагаемое  не зависит от  , а вторая сумма функций зависит от . Поэтому выражение (3.12) можно переписать в виде:



Таким образом, окончательно запишем:

 (1.13)

Данное соотношение справедливо для всех , а для  имеем:

 (1.14)

Полученные рекуррентные соотношения являются основными соотношениями метода динамического программирования. Это так называемые соотношения Беллмана. Они позволяют свести решение задачи нелинейного программирования (1.4)-(1.7) к последовательному решению *N* задач максимизации меньшей размерности.

Соотношение (1.13) позволяет вычислить , если известно , а (3.14) позволяет вычислить максимальное значение целевой функции на последнем этапе. Тогда рекуррентный процесс решения задачи (1.13)-(1.14) должен проводиться в порядке . Действительно, зная  из (3.13) можно найти значения  и т.д.

Рассмотрим алгоритм решения такой задачи.

**1 шаг**. Решается задача (1.14):

.

Решается столько однотипных задач, сколько существует возможных значений фиксированных . Здесь максимизируется функция  по переменной  для каждого возможного фиксированного значения . Решается столько однотипных задач, сколько существует возможных значений фиксированных . Поэтому для каждого  в результате получаются условные точки максимума  и максимальное значение функции  в этих точках.

**2 шаг**. На этом шаге решается задача (1.13) при :



Здесь решаются задачи максимизации функции  по переменной  при каждом возможном фиксированном значении . В результате находятся условные точки максимума  и значения суммы двух функций в этих точках .

Далее, зная , решается задача (1.13) для .

Наконец, при  переходим к шагу *N*:

***N* шаг**. , .

На этом шаге решается только одна задача оптимизации, т.к. - задан. В результате получим точку максимума  и значение .

Для окончательного решение проводим обратное движение алгоритма:

**1 шаг**: ; .

**2 шаг**: ; .

**3 шаг**: ; .

…

**N шаг**: ; .

.

Таким образом, определяется решение исходной задачи нелинейного программирования (1.4)-(1.7) как , при этом максимальное значение целевой функции (4) будет равно значению .

**Достоинства метода динамического программирования**

1. На каждом этапе решается задача поиска экстремума лишь по части переменных, следовательно, размерность этих задач по сравнению с исходной значительно ниже. Это позволяет упростить поиск оптимальных значений искомых переменных.
2. Метод ДП дает возможность решать задачи, которые не могут быть решены другими методами.
3. Алгоритм метода ДП легко реализуется на ЭВМ.

**Недостатки метода динамического программирования**

1. Отсутствие универсального алгоритма, который был бы пригоден для решения всех задач рассматриваемого класса. Алгоритмы ДП объединены лишь общей идеей, и в каждом конкретном случае должны формироваться применительно к специфике прикладной задачи.
2. При большой размерности исходной задачи эти алгоритмы требуют значительных ресурсов ЭВМ.

**1.2. Примеры задач динамического программирования**

Пусть предполагается к осуществлению некоторое мероприятие или серия мероприятий («операция»), преследующая определенную цель. Спрашивается: как нужно организовать (спланировать) операцию для того, чтобы она была наиболее эффективной?

***Задача планирования рабочей силы:***

При выполнении некоторых проектов число рабочих, необходимых для выполнения какого-либо проекта, регулируется путем их найма и увольнения. Поскольку как наем, так и увольнение рабочих связано с дополнительными затратами, необходимо определить, каким образом должна регулироваться численность рабочих в период реализации проекта.

Предположим, что проект будет выполнятся в течение n недель и минимальная потребность в рабочей силе на протяжении *i*-й недели составит *bi* рабочих. При идеальных условиях хотелось бы на протяжении *i*-й недели иметь в точности *bi* рабочих. Однако в зависимости от стоимостных показателей может быть более выгодным отклонение численности рабочей силы как в одну, так и в другую сторону от минимальных потребностей.

Если *xi* – количество работающих на протяжении *i*-й недели, то возможны затраты двух видов:

1) *С1(xi – bi) –* затраты, связанные с необходимостью содержать избыток *xi – bi* рабочей силы;

2) *С2(xi – xi-1)*-затраты, связанные с необходимостью дополнительного найма *(xi – xi-1)* рабочих.

Элементы модели динамического программирования определяются следующим образом:

1. Этап і представляется порядковым номером недели *і*, *і=1,2,…n*.
2. Вариантами решения на *і*-ом этапе являются значения *xi* – количество работающих на протяжении *і*-й недели.
3. Состоянием на *і*-м этапе является *xi-1* – количество работающих на протяжении *(і-1)* –й недели (этапа).

Рекуррентное уравнение динамического программирования представляется в виде



где .

Вычисления начинаются с *n*-го этапа при *xn=bn* и заканчиваются на *1*–ом этапе.

***Задача замены оборудования:***

Чем дольше механизм эксплуатируется, тем выше затраты на его обслуживание и ниже его производительность. Когда срок эксплуатации механизма достигает определенного уровня, может оказаться более выгодной его замена. Задача замены оборудования, таким образом, сводится к определению оптимального срока эксплуатации механизма.

Предположим, что мы занимаемся заменой механизмов на протяжении n лет. В начале каждого года принимается решение либо об эксплуатации механизма еще один год, либо о замене его новым.

Обозначим через *r(t)* и *c(t)* прибыль от эксплуатации *t*-летнего механизма на протяжении года и затраты на его обслуживание за этот же период. Далее пусть *s(t)* – стоимость продажи механизма, который эксплуатировался *t* лет. Стоимость приобретения нового механизма остается неизменной на протяжении всех лет и равна *l*.

Элементы модели динамического программирования таковы:

1. Этап *і* представляется порядковым номером года *і*, *і=1,2,...n*.
2. Вариантами решения на *і*-м этапе (т.е. для *і*-ого года) являются альтернативы: продолжить эксплуатацию или заменить механизм в начале *і*-ого года.
3. Состоянием на *і*-м этапе является срок эксплуатации *t* (возраст) механизма к началу *і*-ого года.

Пусть *fi(t)* – максимальная прибыль, получаемая за годы от *і* до *n* при условии, что в начале *і*-ого года имеется механизм *t*-летнего возраста.

Рекуррентное уравнение имеет следующий вид:



(1) – если эксплуатировать механизм,

(2) – если заменить механизм.

***Задача инвестирования:***

Предположим, что в начале каждого из следующих *n* лет необходимо сделать инвестиции *P1, P2,…, Pn* соответственно. Вы имеете возможность вложить капитал в два банка: первый банк выплачивает годовой сложный процент *r1*, а второй - *r2*. Для поощрения депозитов оба банка выплачивают новым инвесторам премии в виде процента от вложенной суммы.

Премиальные меняются от года к году, и для *і*-ого года равны *qi1*и *qi2* в первом и втором банках соответственно. Они выплачиваются к концу года, на протяжении которого сделан вклад, и могут быть инвестированы в один из двух банков на следующий год. Это значит, что лишь указанные проценты и новые деньги могут быть инвестированы в один из двух банков. Размещенный в банке вклад должен находится там до конца рассматриваемого периода. Необходимо разработать стратегию инвестиции на следующие *n* лет.

Элементы модели динамического программирования следующие:

1. Этап і представляется порядковым номером года і, і=1,2,...n
2. Вариантами решения на *і*-м этапе (для *і*-ого года) являются суммы *li* и  инвестиций в первый и второй банк соответственно.
3. Состоянием *xi* на *і*-м этапе является сумма денег на начало *і*-ого года, которые могут быть инвестированы.

Заметим, что по определению = *xi-li*. Следовательно,



где *і=2,3,…n, x1=P1.* Сумма денег *xi*, которые могут быть инвестированы, включает лишь новые деньги и премиальные проценты за инвестиции, сделанные на протяжении *(і-1)*-го года.

Пусть *fi(xi)* – оптимальная сумма инвестиций для интервала от *і*-го до *n*-го года при условии, что в начале *і*-го года имеется денежная сумма *xi.* Далее обозначим через *si* накопленную сумму к концу *n*-го года при условии, что *li* и *(xi-li)* – объемы инвестиций на протяжении *і*-го года в первый и второй банк соответственно. Обозначая , *і=1,2*, мы можем сформулировать задачу в следующем виде.

Максимизировать *z=s1+s2+…+sn*, где



Так как премиальные за *n*-й год являются частью накопленной денежной суммы от инвестиций, в выражения для *sn* добавлены *qn1* и *qn2*.

Итак, в данном случае рекуррентное уравнение для обратной прогонки в алгоритме динамического программирования имеет вид



где *xi+1* выражается через *xi* в соответствии с приведенной выше формулой, а *fn+1(xn+1)=0*.

**3.3. Задача о загрузке рюкзака (задача о ранце)**

Пусть имеются N видов грузов с номерами .

Единица груза *j*-го вида имеет все *aj*. Если груз j-го вида берется в количестве *xj*, то его ценность в общем случае составляет *F(xj)*. Имеется «рюкзак», грузоподъемность которого равна *B*. Требуется загрузить рюкзак имеющимися грузами таким образом, чтобы вес его был не больше заданного *B*, а ценность «рюкзака» была максимальной.

Составим математическую модель задачи. Пусть *xj* – количество груза *j*-го вида, помещаемого в рюкзак. Тогда можно записать:

 (1.15)

 (1.16)

 (1.17)

Здесь *х*j могут быть и целыми числами. В общем случае это задача нелинейного программирования с сепарабельной целевой функцией, следовательно, она может быть решена методом динамического программирования.

Для этого погрузку «рюкзака» можно интерпретировать как *N*-этапный процесс принятия решений: на 1-м этапе принимается решение о том, сколько нужно взять груза 1-го вида, на 2-ом этапе – сколько груза 2-го вида и т.д. Такая интерпретация наталкивает на возможность применения для решения задачи (1.15) – (1.17) метода динамического программирования. Для этого приведем задачу (1.15) – (1.17) к виду (1.4) – (1.7).

Для этого введем обозначения:  – вес рюкзака перед погрузкой груза *j*-го вида груза или вес рюкзака после погрузки грузов видов 1, 2, …, *j* – 1.Очевидно, что

*y1* = 0. (1.18)

Текущий вес рюкзака определяется выражением

 (1.19)

Текущий вес рюкзака  в силу (3.16) удовлетворяет неравенству

≤ *B*. (1.20)

Очевидно ограничения (1.18) – (1.20) эквивалентны ограничению (1.16), поэтому вместо модели (1.15) – (1.17) можно рассматривать модель (1.15), (1.17) – (1.20). Здесь ограничение (1.20) выводит эту модель за рамки модели (1.4) – (1.7). Для сведения задачи к общему виду задач динамического программирования, запишем (1.20) с учетом (1.19):

.

Отсюда следует: ,

или окончательно с учетом (3.17):

 (1.21)

В результате исходная модель (1.15) – (1.17) свелась к эквивалентной модели вида:

 (1.22)

 (1.23)

 (1.24)

 (1.25)

Задача (3.22)-(3.25) является частным случаем общей задачи динамического программирования, в которой . Здесь ограничение (3.23) является рекуррентным и отражает процесс загрузки рюкзака, а неравенство (3.24) задает область возможных значений .

Рассмотрим решение задачи (3.22)-(3.25) методом динамического программирования:

1 шаг. Вычисляется величина

 (1.26)

В результате решения серии задач максимизации получаем точки максимума   и значения .

*S*-тый шаг (). Вычисляются величины

 (1.27)

В результате решения серии задач максимизации, получаем  и . При *s=1* решается только одна задача на максимум, т.к. значение - задано.

Для определения безусловных точек максимума, т.е. решения исходной задачи, проводим обратное движение алгоритма:

.

Отсюда:

.

Далее: . И так далее . Причем  есть максимальное значение целевой функции.

Наличие условия целочисленности переменных *xj* и  упрощает решение задачи. В этом случае . Здесь [*В*] указывает на то, что берется целая часть числа. Если  не целые, то .

**Пример:**

Имеется свободный капитал в размере 4 млн. у.е. Этот капитал может быть распределен между 4-мя предприятиями, причем распределение осуществляется только целыми частями (0, 1, 2, 3 или 4 млн. у.е.). Прибыль, получаемая каждым предприятием при инвестировании в него определенной суммы, указана в таблице.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Предпр.  Капитал | 0 млн. у.е. | 1 млн. у.е. | 2 млн. у.е. | 3 млн. у.е. | 4 млн. у.е. |
| 1-е предпр. | 0 | 10 | 17 | 25 | 36 |
| 2-е предпр. | 0 | 11 | 16 | 25 | 35 |
| 3-е предпр. | 0 | 10 | 18 | 24 | 34 |
| 4-е предпр. | 0 | 9 | 19 | 26 | 35 |

Требуется распределить инвестиции между предприятиями из условия максимальной общей прибыли.

Обозначим: *хj*- количество капиталовложений, выделенных *j*-тому предприятию (). Тогда прибыль, записанная в таблице, можно обозначить как *Fj(xj)* (). Например, F1(0)=0; F1(1)=10; F1(2)=17 и т.д. .... F2(0)=0; F2(1)=11; F4(4)=35.

Тогда математическая модель примет вид:





*хj≥*0 – целые, ()

Данная модель является частным случаем задачи о загрузке рюкзака, где *N=4, В=4, аj=1* (). Введя новую переменную *yj*- израсходованные средства до выделения капиталовложений *j*-тому предприятию, приведем исходную модель к виду задачи динамического программирования:



; ()

*y1=*0;

; ()

Решение задачи проведем в соответствии с алгоритмом динамического программирования:

**1 шаг.**



1. Зафиксируем *y4=0*. Тогда допустимые значения *x4∈[0, 4-0] =[0,1,2,3,4]*.
   1. *x4=0*. Тогда *F4(0)=0*.
   2. *x4=1*. *F4(1)=9*.
   3. *x4=2*. *F4(2)=19*.
   4. *x4=3*. *F4(3)=26*
   5. *x4=4*. *F4(4)=35*.

Максимальное значение , и достигается оно при *x4=4*. Таким образом, заполняется первая строчка таблицы.

1. Зафиксируем *y4=1*. Тогда допустимые значения *x4∈[0, 4-1]= [0,1,2,3].*
   1. *x4=0*. Тогда *F4(0)=0*.

2.2) *x4=1*. *F4(1)=9*.

2.3) *x4=2*. *F4(2)=19*.

2.4) *x4=3*. *F4(3)=26.*

Максимальное значение , и достигается оно при *x4=3*. Таким образом, заполняется вторая строка таблицы.

Далее аналогично фиксируем *y4=2, y4=3, y4=4*. Заполняем оставшиеся строки таблицы.

*Таблица шага №1*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| y4 x4 | 0 | 1 | 2 | 3 | 4 |  |  |
| 0 | 0 | 9 | 19 | 26 | 35 | 35 | 4 |
| 1 | 0 | 9 | 19 | 26 | - | 26 | 3 |
| 2 | 0 | 9 | 19 | - | - | 19 | 2 |
| 3 | 0 | 9 | - | - | - | 9 | 1 |
| 4 | 0 | - | - | - | - | 0 | 0 |

**2 шаг.**



1. Зафиксируем *y3=0*. Тогда допустимые значения *x3∈[0, 4-0] =[0,1,2,3,4]*.
   1. *x3=0*. Тогда *F3(0)=0*. Определим значение второго слагаемого:  при *y3=0* и *x3=0*. Найдем *y4=0+0=0*. Тогда, обратившись к таблице шага 1, увидим, что **. Следовательно, *F3(0)+ =0+35=35*. Этот результат заносим в таблицу шага 2 в ячейку, соответствующую *y3=0* и *x3=0*.
   2. *x3=1*. Аналогично: *F3(1)=10*. Найдем *y4= y3+ x3=0+1=1*. Из таблицы шага 1 определим: =. Сумма *F3(1)+ =10+26=36*.
   3. *x3=2*. *F3(2)=18*. *y4=0+2=2*. ⇒ ==*19*. Тогда *F3(2)+ =18+19=37*.
   4. *x3=3*. *F3(3)=24*, *y4=0+3=3*. ⇒ ==*9*. Тогда *F3(3)+ =24+9=33*.
   5. *x3=4. F3(4)=34. y4=0+4=4. ⇒ ==0*. Тогда *F3(4)+  =34+0=34*.

Максимальное значение =37, и достигается оно при *x3=2*. Первая строчка таблицы заполнена.

1. Зафиксируем *y3=1*. Тогда допустимые значения *x3∈[0, 4-1]= [0,1,2,3]*.
   1. *x3=0. F3(0)=0. y4=1+0=1. ⇒ ==26*. Тогда *F3(0)+  =0+26=26*.
   2. *x3=1. F3(1)=10. y4=1+1=2. ⇒ == =19*. Тогда *F3(1)+  =10+19=29*.
   3. *x3=2. F3(2)=18. y4=1+2=3. ⇒ ==9*. Тогда F3(2)+  =18+9=27.
   4. *x3=3. F3(3)=24 y4=1+3=4. ⇒ ==0*. Тогда *F3(3)+  =24+0=24*.

Максимальное значение , и достигается оно при *x3=1*. Таким образом, заполняется вторая строка таблицы.

1. Зафиксируем *y3=2*. Тогда допустимые значения *x3∈[0, 4-2]= [0,1,2]*.

3.1) *x3=0. F3(0)=0. y4=2+0=2. ⇒ f4(2) =19. F3(0)+ f4(2)=0+19=19*.

* 1. *x3=1. F3(1)=10. y4=2+1=3. ⇒ =9. F3(1)+  =10+9=19*.
  2. *x3=2. F3(2)=18. y4=2+2=3. ⇒ =0. F3(2)+  =18+0=18*.

Максимальное значение  достигается при двух возможных значениях *x3: x3=1 и x3=0*. В таблицу можно занести любое из них. Таким образом, заполняется третья строка таблицы.

Далее аналогично фиксируем *y3=3, y3=4*. Заполняем оставшиеся строки таблицы.

*Таблица шага №2*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| y3 x3 | 0 | 1 | 2 | 3 | 4 |  |  |
| 0 | 35 | 36 | 37 | 33 | 34 | 37 | 2 |
| 1 | 26 | 29 | 27 | 24 | - | 29 | 1 |
| 2 | 19 | 19 | 18 | - | - | 19 | 0 (или 1) |
| 3 | 9 | 10 | - | - | - | 10 | 1 |
| 4 | 0 | - | - | - | - | 0 | 0 |

**3 шаг.**



Все вычисления производятся аналогично шагу 2. Не останавливаясь более подробно на этапах решения подзадачи данного шага, приведем получившуюся в результате таблицу.

*Таблица шага №3*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| y2 x2 | 0 | 1 | 2 | 3 | 4 |  |  |
| 0 | 37 | 40 | 35 | 35 | 35 | 40 | 1 |
| 1 | 29 | 30 | 26 | 25 | - | 30 | 1 |
| 2 | 19 | 21 | 16 | - | - | 21 | 1 |
| 3 | 10 | 11 | - | - | - | 11 | 1 |
| 4 | 0 | - | - | - | - | 0 | 0 |

**4 шаг.**



Последний шаг интересен тем, что здесь решается единственная задача максимизации при заданном *y1=0*.

*y1=0*. Следовательно *x1∈[0, 4-0]= [0,1,2,3,4]*. Выполняя все действия, аналогично предыдущим шагам, получим таблицу последнего шага, состоящую из единственной строки, соответствующей *y1=0*.

*Таблица шага №4*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| y1 x1 | 0 | 1 | 2 | 3 | 4 |  |  |
| 0 | 40 | 40 | 38 | 36 | 36 | 40 | 0 (или 1) |

Далее проводим обратное движение алгоритма:

1. *y1=0, x1\*=0, ⇒ y2\*= y1+ x1\*=0+0=0*.
2. Определяем значение x2\* из таблицы шага № 3 по найденному *y2\*=0*. Значению *y2= y2\*=0* соответствует значение *x2(y2)=1*. Следовательно, *x2\*=1*. Далее можно определить *y3\*= y2\*+ x2\*=0+1=1*.
3. Аналогично, обращаясь к таблице шага №2, найдем: *x3\*= x3(1)=1, ⇒ y4\*= y3\*+ x3\*=1+1=2*.
4. Из таблицы шага №1 : *x4\*= x4(2)=2*.

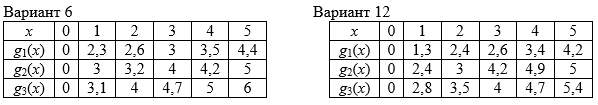
Окончательно имеем: первому предприятию средства не выделяются (*x1\*=0*), второму выделяется 1 млн. у.е. (*x2\*=1*), третьему предприятию – 1 млн. у.е. (*x3\*=1*), и четвертому – 2 млн. у.е. (*x4\*=2*). При этом значение целевой функции (общая прибыль по всем 4-м предприятиям) составит:

==40.

**ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ**

**1 задание**

****

****

**2 задание**

****

**Контрольные вопросы**

1.В чем суть принципа оптимальности Беллмана.

2.Основные примеры задач динамического программирования.

3.Что означает реккурентность вычислений?