

Динамическое программирование

Немного теории

Динамическое программирование — метод решения сложных задач путём разбиения их на более простые подзадачи.

- Применяется к задачам, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной.
- В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Это уже известно...

Последовательность Фибоначчи:

- $F_n = F_{n-1} + F_{n-2}$
 - Каждое число в последовательности вычисляется как сумма двух предыдущих
- «Наивным» будет решение «в лоб», через рекурсию...

Числа Фибоначчи

```
int fib(int n) {  
    if (n<2) return 1;  
    return fib(n-1)+fib(n-2);  
}
```

Числа Фибоначчи

```
int fib(int n) {  
    if (n<2) return 1;  
    return fib(n-1)+fib(n-2);  
}
```

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

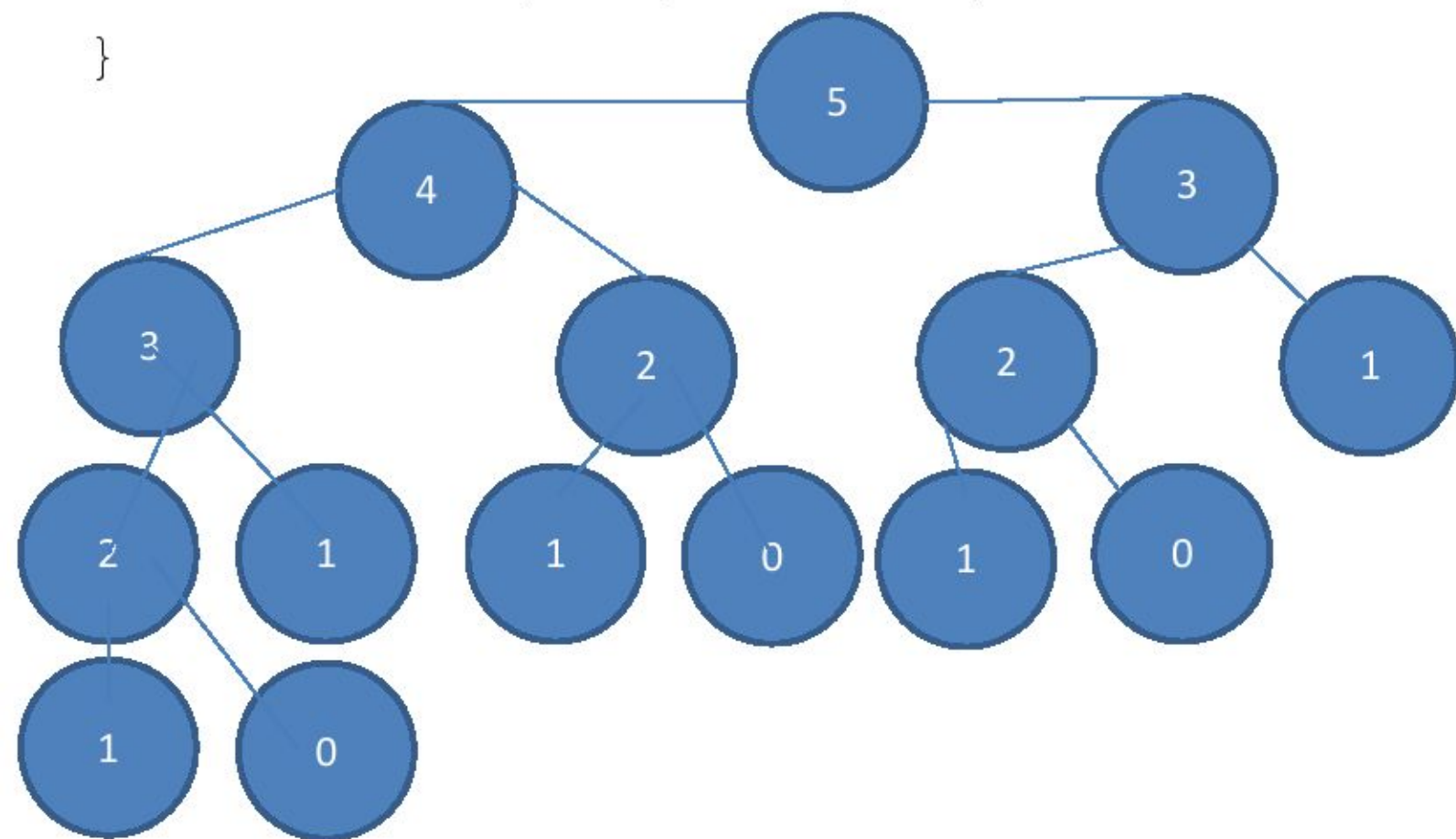
$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$

$\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$

Числа Фибоначчи

```
int fib(int n) {  
    if (n<2) return 1;  
    return fib(n-1)+fib(n-2);  
}
```



Числа Фибоначчи

Решение методом «динамического программирования» предполагает запоминание каждого числа в массиве.

Тогда N-е число Фибоначчи легко можно найти по следующей формуле:

$$F[N] = F[N-1] + F[N-2], \text{ при } N > 2.$$

Задача об n – битных двоичных числах

Найти количество F всех n – битных двоичных чисел, у которых в двоичной записи нет подряд идущих двух единиц. ($N \leq 30$).

Задача об n – битных двоичных числах

N(кол-во бит)	Числа, удовлетворяющие условию задачи	F(N)
1	0,1	F(1)=2
2	00,01,10, 11	F(2)=3
3	000,001,010, 011 , 100,101, 110 , 111	F(3)=5

$$F[N] = F[N-1] + F[N-2], \text{ при } N > 2.$$

Зайчик на лесенке

На вершине лесенки, содержащей N ступенек, находится зайчик, который начинает прыгать по ним вниз, к основанию.

Зайчик может прыгнуть на следующую ступеньку, на ступеньку через 1 или 2.

Определить число всевозможных “маршрутов” зайчика с вершины на землю.

Зайчик на лесенке

Пусть зайчик находится на ступеньке с номером X .

По условию он может прыгнуть на ступеньки с номерами $X - 1$, $X - 2$ и $X - 3$.

Пусть $F(X)$ - число маршрутов со ступеньки X до земли

$F(1)$	1
$F(2)$	1+1, 2
$F(3)$	1+1+1, 1+2, 2+1, 3

$$F[X] = F[X - 1] + F[X - 2] + F[X - 3]$$

Программа на C++

```
#include <iostream>
using namespace std;
int main()
{   int N; long long F[31];
    cin>>N;
    F[1]=1;F[2]=2;F[3]=4;
    for(int i=4; i <= N; i++)
        F[i]=F[i-1]+F[i-2]+F[i-3];
    cout<<F[N];
    return 0;
}
```

Задача о фишке

Фишка может двигаться по полю длины N только вперед. Длина хода фишки не более K ($N, K \leq 10$).

Найти количество различных путей, по которым фишка может пройти поле от позиции 1 до позиции N .

Задача о фишке

Пусть $S[i]$ - количество различных путей, по которым фишка может пройти поле от начала до позиции с номером i . Предположим, что для любого j от 1 до i известны значения величин $S[j]$. Задача состоит в определении правила вычисления значения $S[i+1]$, используя значения известных величин. Заметим, что в позицию с номером $i+1$ фишка может попасть из позиций $i, i-1, \dots, i-k$.

$$S[i+1] = S[i] + S[i-1] + \dots + S[i-k]$$

Вычисляя последовательно значения величин $S[1], S[2], \dots, S[N]$ по правилу, получаем значение $S[N]$ – решение задачи

Алгоритм динамического программирования

Динамическое программирование — метод оптимизации, приспособленный к задачам, в которых требуется построить решение с максимизацией или минимизацией, т.е. оптимальным значением некоторого параметра.

Его алгоритм можно сформулировать так:

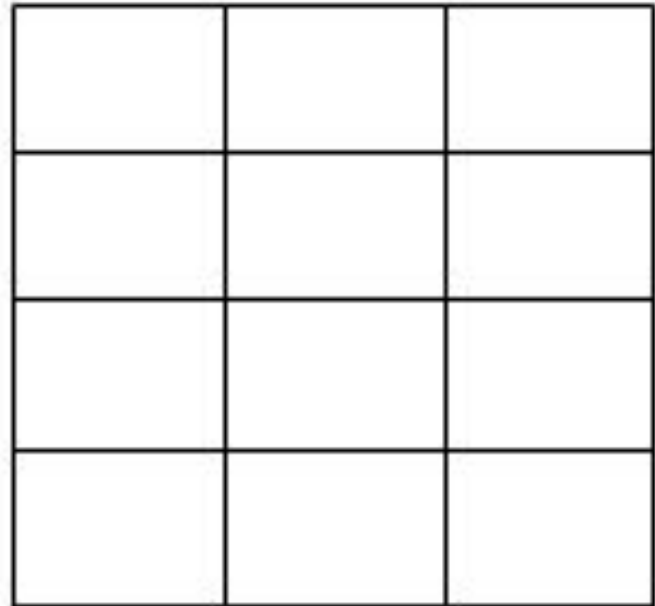
1. Выделить и описать подзадачи, через решение которых будет выражаться искомое решение;
2. Выписать рекуррентные соотношения (уравнения), связывающие оптимальные значения параметра для всех подзадач;
3. Вычислить оптимальное значение параметра для всех подзадач;
4. Построить само оптимальное решение.

В задачах на подсчет количеств допустимых вариантов (задачи рассмотрены выше) пункт 4 не нужен



Задача о черепашке

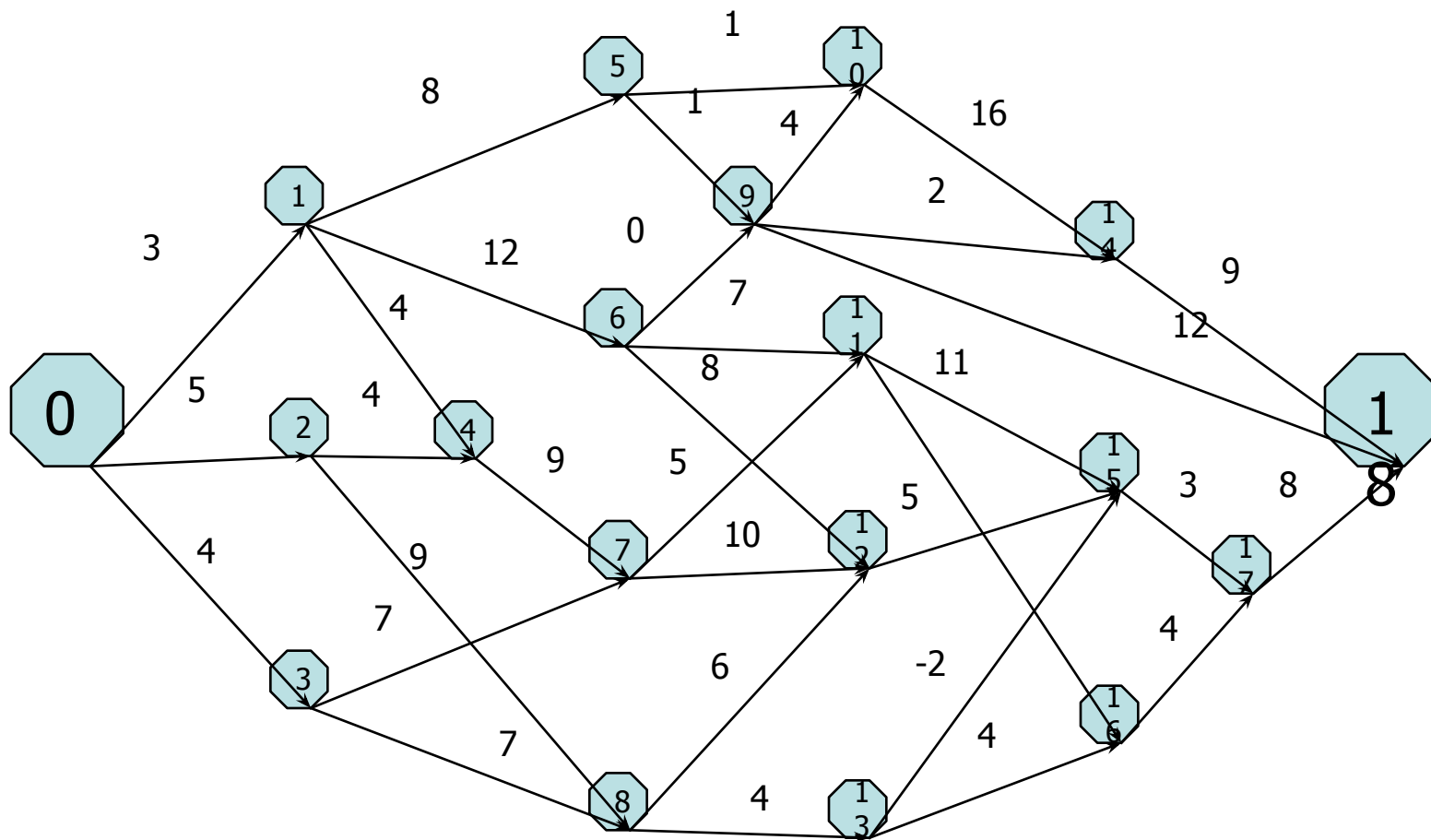
Сколько вариантов пройти с левого нижнего угла в правый верхний угол?



Формулировка задачи динамического программирования

- Дано:
 - множество состояний
 - в том числе *начальное* и *конечное*
 - множество возможных переходов из одного состояния в другое
 - с каждым переходом связывается числовой параметр
 - интерпретируется как затраты, выгода, расстояние, время и т. п.
- Найти:
 - оптимальную последовательность переходов (*путь*) из начального состояния в конечное
 - максимум или минимум суммы числовых параметров
 - предполагается, что хотя бы один путь из начального состояния в конечное существует

Пример



Математическая запись

$$\max_{(x_{ij} | i \in I, j \in J)} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\sum_{i \in I} x_{ij} \leq 1, j \in J \setminus \{\# J\}$$

$$\sum_{i \in I} x_{ij} = \sum_{k \in J} x_{jk}, j \in J \setminus \{\# J\}$$

$$x_{ij} \in \{0;1\}, i \in I, j \in J \setminus \{\# J\}, (i, j) \in A$$

$$x_{ij} = 0, i \in I, j \in J \setminus \{\# J\}, (i, j) \notin A$$

x_{ij} переменная включения дуги (i, j) в путь

c_{ij} числовое значение, приписанное дуге

I множество начальных вершин дуг

J множество конечных вершин дуг

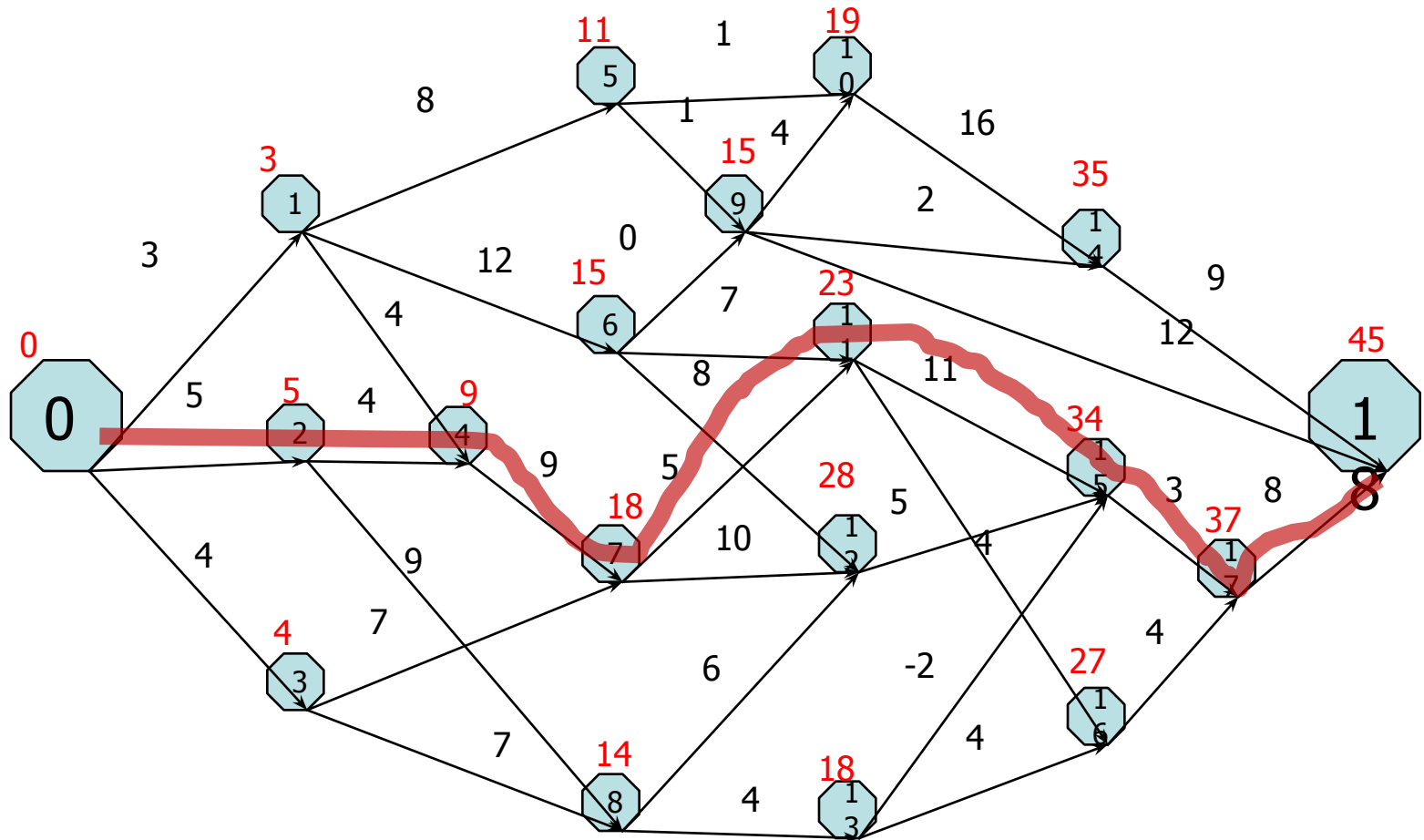
A множество всех дуг

$\#$ оператор «число элементов во множестве»

Принцип оптимальности Беллмана

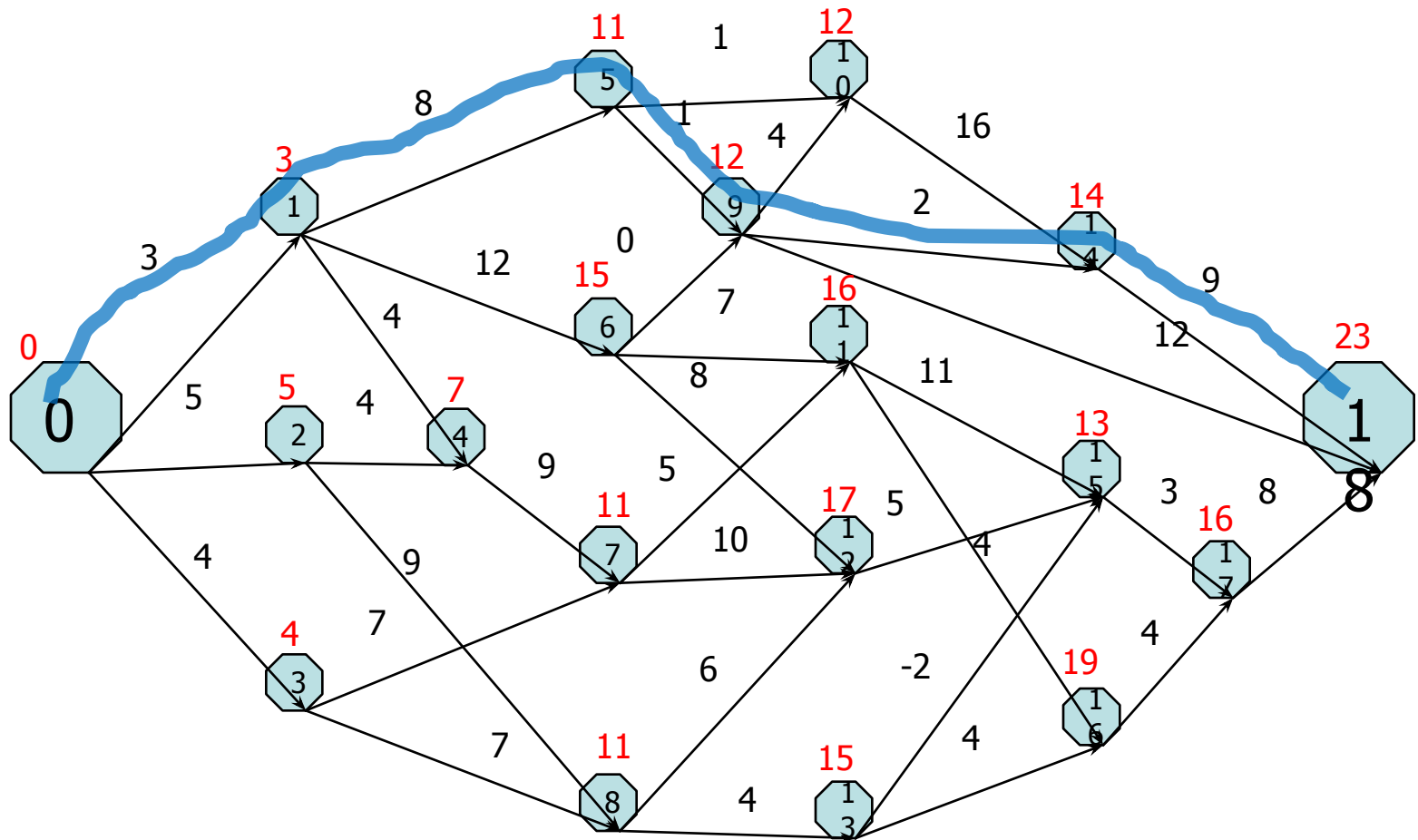
- Если вершины A и B лежат на оптимальном пути между вершинами O и X , то часть оптимального пути от O до X между вершинами A и B непременно является оптимальным путём от A до B .
- Следствие
 - Чтобы найти оптимальный путь от O до A , достаточно исследовать продолжения к A всех оптимальных путей до вершин, предшествующих A
 - Продолжения неоптимальных путей к предшествующим вершинам можно не просчитывать: они никогда не дадут оптимального пути к A
- Принцип Беллмана позволяет построить простую и эффективную вычислительную процедуру для решения задач динамического программирования

Алгоритм решения задач динамического программирования



максимум

Алгоритм решения задач динамического программирования



МИНИМУМ

Экономические приложения

Управление проектами

- Дуги - работы, которые должны быть выполнены
- Параметры – продолжительность работ
- Самый длинный путь определяет минимальный срок выполнения проекта

Управление реновацией основных средств производства

- Дуги соответствуют решениям:
 - *e.g.*, эксплуатировать; ремонтировать; списать
- Параметры –доходы
- Самый длинный путь определяет жизненный цикл элемента основных средств

Бизнес-планирование

- Дуги – операции, переводящие фирму в новое состояние
- Параметры – доходы (расходы)
- Самый длинный путь определяет наилучший бизнес-план

Поиск оптимального маршрута

- Дуги – пути сообщения
- Параметры – время (или стоимость) перевозки
- Самый короткий (дешёвый) путь определяет оптимальную перевозку

Условия применения динамического программирования

1. Оптимальное решение задачи выражается через оптимальное решение задач меньшей размерности, представимых в виде подзадач (подпрограмм). Улучшая решение подзадач, тем самым, улучшается решение общей задачи.

2. Небольшое число подзадач, что позволяет хранить решения каждой подзадачи в таблице. Уменьшение числа подзадач происходит из-за многократного их повторения(т.н. перекрывающиеся подзадачи).

3. Дискретность (неделимость) величин, рассматриваемых в задаче.

4. Один из главных критериев, когда принцип ДП дает эффект уменьшения временной сложности: если в процессе решения необходимо многократно перебирать одни и те же варианты (за счет увеличения емкостной сложности уменьшается временная сложность).