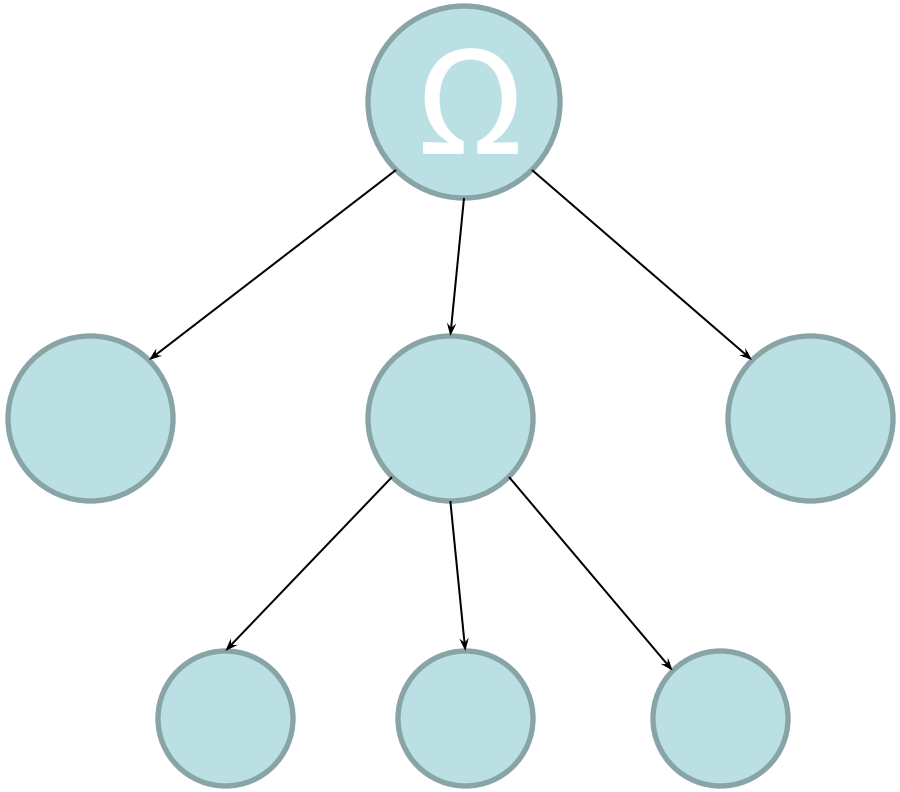
§ 3. Динамическое программирование

Динамическое программирование – это процесс пошагового решения задач, когда на каждом шаге из множества допустимых решений выбирается одно решение, оптимизирующее целевую функцию.

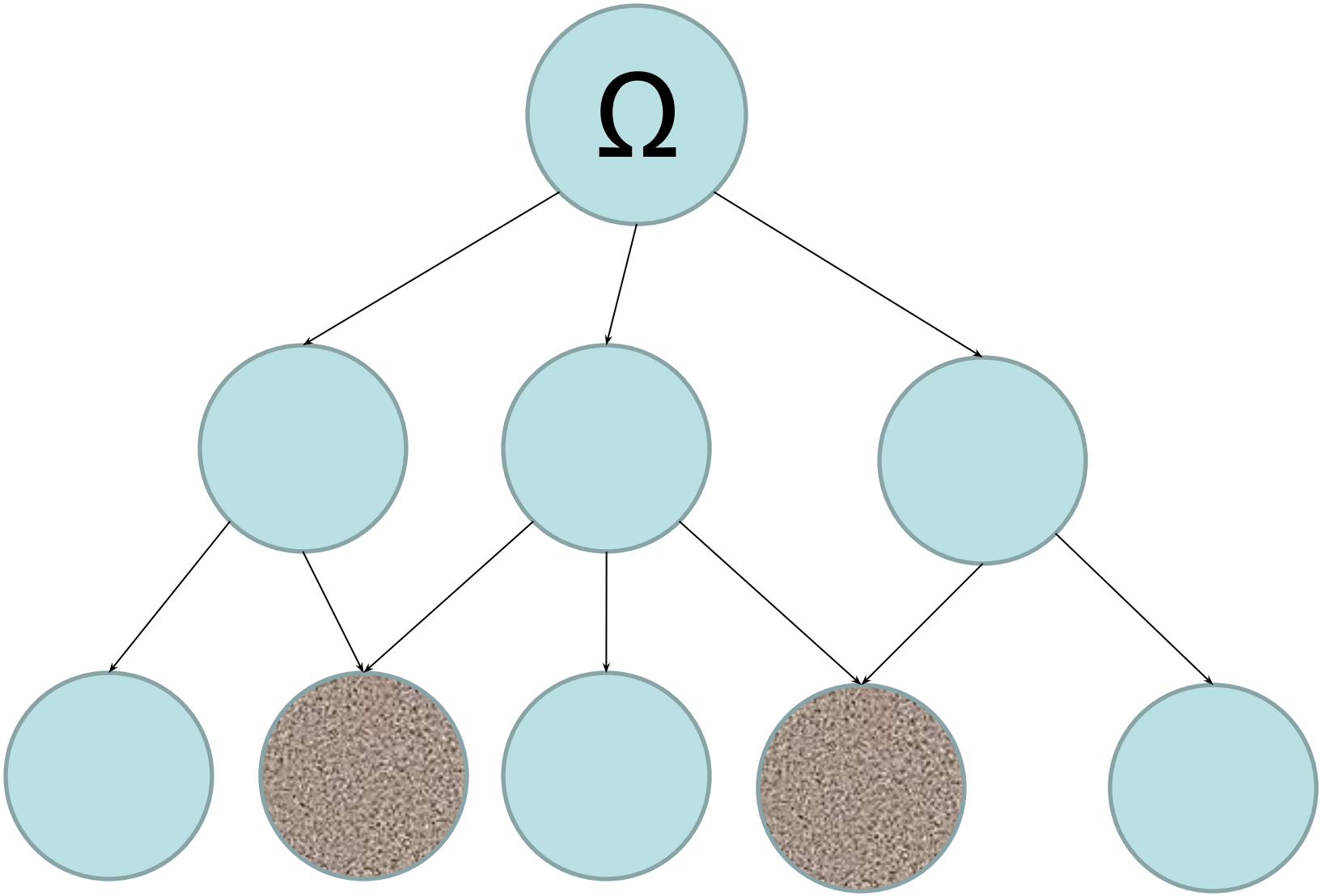
Динамическое программирование применимо для задач, обладающих следующими свойствами:

1. Свойство оптимальности для подзадач.
2. Наличие перекрывающихся подзадач.

Разбиение на подзадачи в методе ветвей и границ:



Наличие перекрывающихся подзадач:



При использовании динамического программирования каждая из подзадач решается только один раз и ее решение запоминается в специальной таблице.

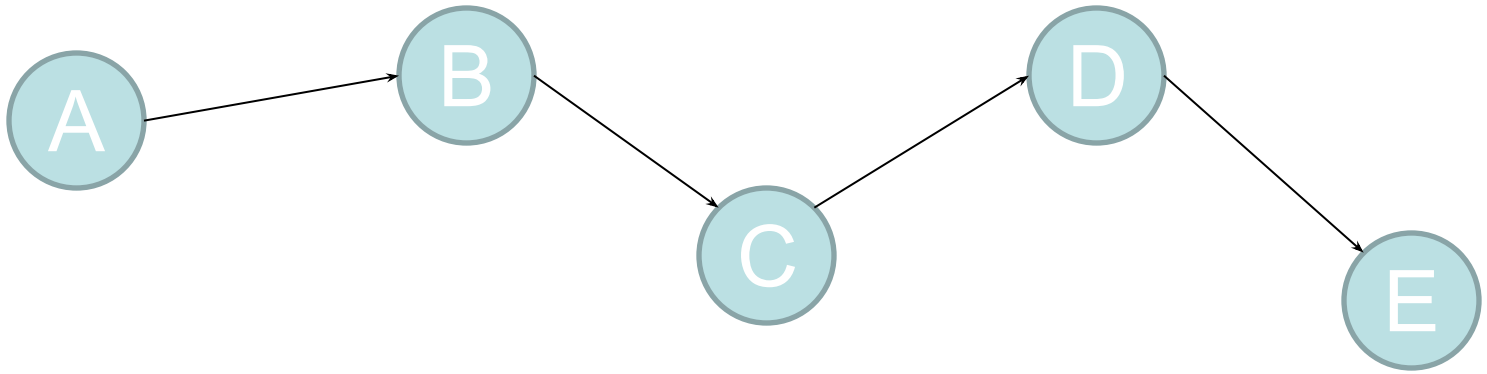
При повторном появлении подзадачи она не решается, а ответ берется из таблицы.

Принцип оптимальности для подзадач.

Говорят, что для задачи выполнен принцип оптимальности для подзадач, если

оптимальное решение задачи содержит оптимальные решения подзадач.

Пример – кратчайший путь в графе. Каждый фрагмент кратчайшего пути также является кратчайшим путем.



Задача о перемножении матриц.

Дано n матриц M_1, M_2, …, M_n

Матрица M_i имеет размеры p_{i-1} на p_i

Для перемножения матрицы A размером m x k

на матрицу B размером k x n требует mkn операций

C = AB

$$C_{ij} = \sum_{t=1}^{k} A_{it} B_{tj}$$

Необходимо расставить скобки в произведении

$M_1 M_2 \ldots M_n$

то есть требуется найти порядок перемножения матриц, при котором общее число операций минимально.

Пример:

M_1 10 x 100

M_2 100 x 5

M_3 5 x 50

$((M_1 M_2) M_3)$

$M_1 * M_2$ 10*100*5 = 5000

$* M_3$ 10*5*50 = 2500

сумма = 7500

$(M_1 (M_2 M_3))$

$M_2 * M_3$ 100*5*50 = 25000

$M_1 *$ 10*100*50 = 50000

сумма = 75000

Обозначим $M_{i...j} = M_i M_{i+1} \ldots M_j$

$M_{i...j} = (M_i \ldots M_k)*(M_{k+1} \ldots M_j)$ для некоторого $i<=k<j$

f_{ij} – минимальное число операций для вычисления $M_{i...j}$

Нас интересует f_{1n}

Очевидно, что $f_{ii} = 0$

Рекуррентное соотношение:

$f_{ij} = \min_k \{f_{ik} + f_{k+1j} + p_{i-1}\, p_k\, p_j\}, \; i < j$

Для запоминания порядка перемножения:

$g_{ij} = \operatorname{argmin}_k \{f_{ik} + f_{k+1j} + p_{i-1}\, p_k\, p_j\}$

Псевдокод алгоритма:

```
for i = 1 to n do
   f_ii = 0
i = 1, j = 2, t = 2
while j - i < n do
   f_ij = ∞
   for k = i to j - 1 do
      t = f_ik+f_k+1j+p_i-1p_kp_j
      if t < f_ii then
         f_ij = t, g_ij = k
   i++, j++
   if j = n then
      i = 1, t++, j = t
```

Перемножение матриц:

```
MULT (i, j)
   if j > i then
      X = MULT (i, g_ij)
      Y = MULT (g_ij+1, j)
      return XY
   else
      return M_i
```

Запуск:

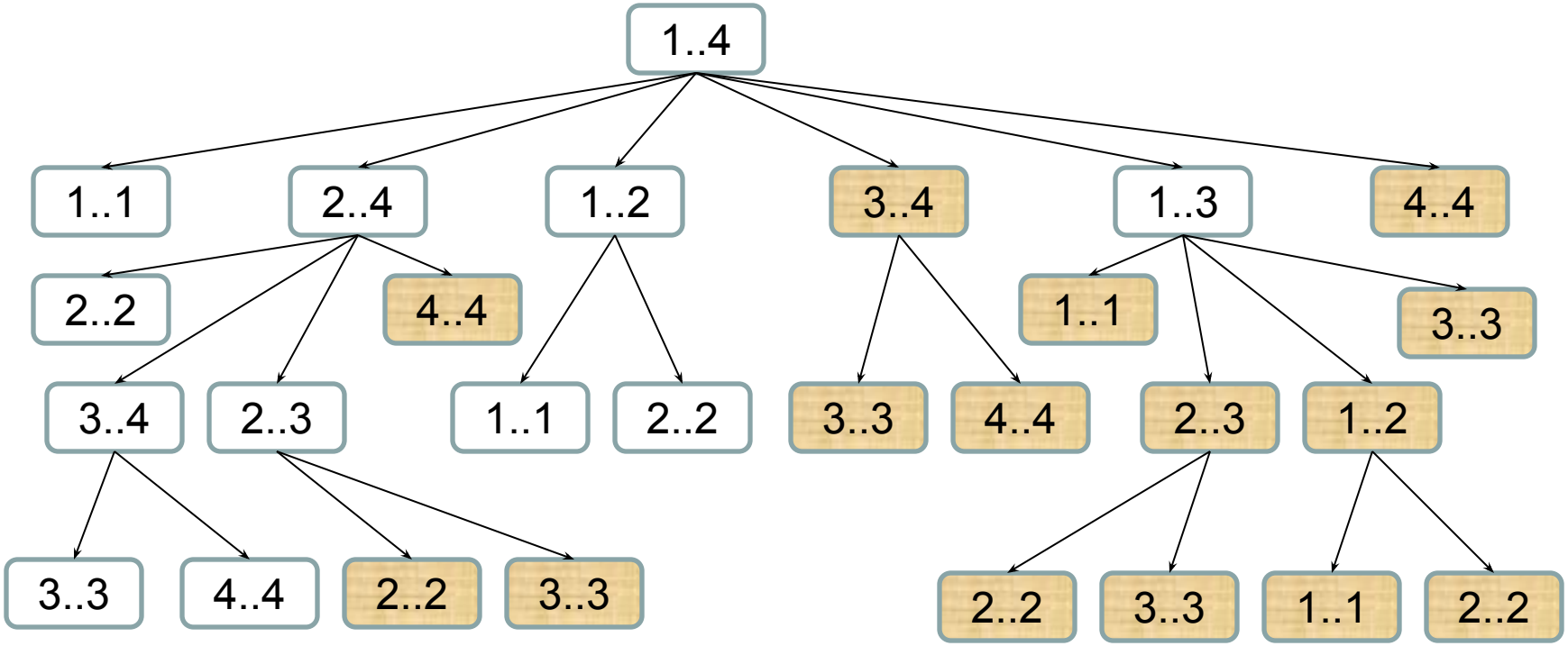
```
MULT (1, n)
```

Численный пример:

n = 4

Матрица	Число строк	Число столбцов
M_1	2	4
M_2	4	3
M_3	3	2
M_4	2	1

Разбиение на подзадачи:



Динамическое программирование:

- сверху вниз;
- снизу вверх.

Задачи, для которых применимо динамическое программирование:

1. Задача о линейном раскрое.
2. Задача о рюкзаке.
3. Задача о наибольшей общей подпоследовательности.
4. Задача поиска самой длинной неубывающей подпоследовательности.