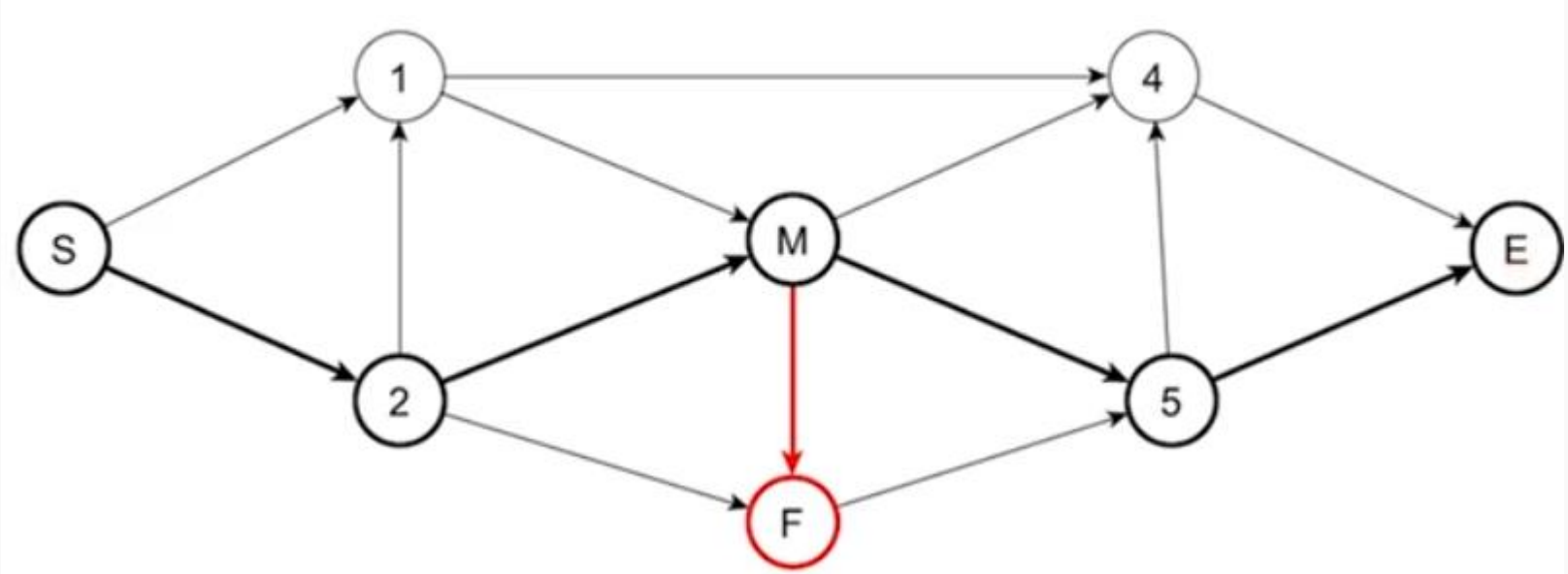
ЛЕКЦИЯ 13
ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ЛЕКТОР ФУРМАВНИН С.А.

ПРИНЦИП ОПТИМАЛЬНОСТИ БЕЛЛМАНА



- Если траектория от S до E оптимальна, то и траектория от S до M тоже.
- Но не наоборот. Траектории от S до M и от M до F вместе не дают оптимальную траекторию

ФУНКЦИОНАЛЬНОЕ УРАВНЕНИЕ БЕЛЛМАНА

- «*An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision*» [Bellman]
- Изменение состояния: $x_t \to x_{t+1} = T(x_t, a_t)\ with\ pay-off\ F(x_t, a_t)$
- Обозначим $V(x_0) = \max\limits_{a_0 \dots a_T} \sum_t F(x_t, a_t)$
- Тогда $V(x_t) = \max\limits_{a_t} \{F(x_t, a_t) + V(T(x_t, a_t))\}$
- Простыми словами: любой кусок оптимальной траектории является оптимальной траекторией

ЗАДАЧА РАЗМЕНА МОНЕТ

- Пусть есть монеты номиналами $c_1 \dots c_k$ и необходимо разменять сумму N, взяв наименьшее количество монет
- x_t - это оставшаяся сумма для размена
- $a_t \in \{c_1 \dots c_k\}$ - это решение сколько мы берём на каждом шаге

$x_{t+1} = x_t - a_t$

$F(x_t, a_t) = 1$

- мы должны минимизировать количество шагов, т.е. задача на минимизацию, а не на максимизацию
- Как записать функцию Беллмана?

ЗАДАЧА РАЗМЕНА МОНЕТ

- Пусть есть монеты номиналами $c_1 \dots c_k$ и необходимо разменять сумму N, взяв наименьшее количество монет
- Запишем функциональное уравнение Беллмана
- $V(t) = \min\big(V(t - c_1), \dots, V(t - c_k)\big) + 1$
- Иными словами: на каждом шаге мы берем такую монету, чтобы функция Беллмана была минимальной на прошлом шаге.
- Нас не волнует то, что тут min , так как $\max\big(f(t)\big) = -\min\big(-f(t)\big)$ и наоборот

ЗАДАЧА РАЗМЕНА МОНЕТ

- Приложение к размену монет: пусть есть монеты номиналами c_1, c_2, c_3 и необходимо разменять сумму N
- Заведем таблицу $V(x_k)$ от 0 до N и протабулируем оптимальные размены. Пусть номиналы 1, 3, 4 и надо разменять 1, используя минимальное количество монет

k	1	2	3	4	5	6	7	8	9	10

V	1	2								

ЗАДАЧА РАЗМЕНА МОНЕТ

- Приложение к размену монет: пусть есть монеты номиналами c_1, c_2, c_3 и необходимо разменять сумму N
- Заведем таблицу $V(x_k)$ от 0 до N и протабулируем оптимальные размены. Пусть номиналы 1, 3, 4 и надо разменять 1, используя минимальное количество монет

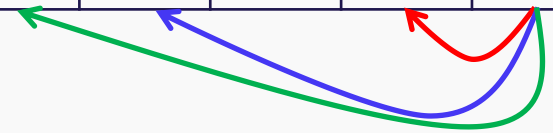
k	1	2	3	4	5	6	7	8	9	10

V	1	2	1	1						

ЗАДАЧА РАЗМЕНА МОНЕТ

- Приложение к размену монет: пусть есть монеты номиналами c_1, c_2, c_3 и необходимо разменять сумму N
- Заведем таблицу $V(x_k)$ от 0 до N и протабулируем оптимальные размены. Пусть номиналы 1, 3, 4 и надо разменять 1, используя минимальное количество монет

k	1	2	3	4	5	6	7	8	9	10
V	1	2	1	1	2					



ЗАДАЧА РАЗМЕНА МОНЕТ

- Приложение к размену монет: пусть есть монеты номиналами c_1, c_2, c_3 и необходимо разменять сумму N
- Заведем таблицу $V(x_k)$ от 0 до N и протабулируем оптимальные размены. Пусть номиналы 1, 3, 4 и надо разменять 1, используя минимальное количество монет

k	1	2	3	4	5	6	7	8	9	10
V	1	2	1	1	2	2	2	2	3	3

ВОСХОДЯЩЕЕ И НИСХОДЯЩЕЕ РЕШЕНИЕ

```
int calc_change(int n, int m, int *changes, int *V) {
  int i;
  if (n == 0) return 0;
  if (n < 0) return INT_MAX;
  if (V[n] != INT_MAX return V[n];
  for (i = 0; i < m; ++i) {
    int CT = calc_change(n - changes[i], m, changes, V);
    if (CT == INT_MAX) continue;
    V[n] = min(V[n], CT + 1);
  }
  return V[n];
}
```

$$V(t) = \min\left(V(t - c_1), \dots, V(t - c_k)\right) + 1$$

ЗАДАЧА О РЮКЗАКЕ

Вам необходимо написать программу, которая, считывая со стандартного ввода:

- количество вещей
- общий вес, входящий в рюкзак
- Вес каждой вещи w_i
- Стоимость каждой вещи v_i

выдавало бы наибольшую стоимость вещей, которые можно положить в рюкзак.

Например, для 4 6 4 3 2 1 5 4 3 2 ответом будет 2 + 3 + 4 = 9

ФУНКЦИОНАЛЬНОЕ УРАВНЕНИЕ БЕЛЛМАНА

$$V(0, w) = 0$$
$$V(i, w) = V(i-1, w) \text{ для } w < w_i$$
$$V(i, w) = \max(V(i-1, w), V(i-1, w-w_i) + v_i) \text{ для } w > w_i$$

i	v	w
1	5	4
2	4	3
3	3	2
4	2	1

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	5	5	5
2	0	0	0	4	5	5	5
3	0	0	3	4	5	7	8
4	0	2	3	5	6	7	9

ОБСУЖДЕНИЕ

- Довольно сложно было догадаться в задаче о рюкзаке, что функция Беллмана будет двумерной
- У нас там две степени свободы: каждую вещь можно взять или не взять (первый уровень) с учетом того, как взяты остальные вещи (второй уровень)

РАССТОЯНИЕ РЕДАКТИРОВАНИЯ

Как перейти от слова **spoon** к слову **sponge**?

Что можно делать:

- Вставлять букву в любое место (`cost == 1`)
- Удалять любую букву (`cost == 1`)
- Изменять любую букву (`cost == 2`)

Нужно найти последовательность действий с минимальной ценой

spoon → sponn (+2) → spong (+2) → sponge (+1), cost == 5

spoon → spon (+1) → spong (+1) → sponge (+1), cost == 3

Можем ли мы применить динамическое программирование?

ФУНКЦИОНАЛЬНОЕ УРАВНЕНИЕ БЕЛЛМАНА

Любая часть оптимальной траектории является оптимальной траекторией

$$V(i,j) = \min \begin{cases} V(i-1,j)+1 \\ V(i,j-1)+1 \\ V(i-1,j-1)+K(i,j) \end{cases}$$

$$K(i,j) = \begin{cases} 0, S_1(i) = S_2(j) \\ 2, S_1(i) \neq S_2(j) \end{cases}$$

Теперь задача сводится к тому, чтобы найти $V(5,6)$

	#	S	P	O	N	G	E
#	0	1	2	3	4	5	6
S	1	$V(1,1)$					
P	2						
O	3						
O	4						
N	5						$V(5,6)$

Потренируемся заполнять таблицу?

ФУНКЦИОНАЛЬНОЕ УРАВНЕНИЕ БЕЛЛМАНА

Любая часть оптимальной траектории является оптимальной траекторией

$$V(i,j) = \min \begin{cases} V(i-1,j)+1 \\ V(i,j-1)+1 \\ V(i-1,j-1)+K(i,j) \end{cases}$$

$$K(i,j) = \begin{cases} 0, S_1(i) = S_2(j) \\ 2, S_1(i) \neq S_2(j) \end{cases}$$

Теперь задача сводится к тому, чтобы найти $V(5,6)$

Нашли!

	#	S	P	O	N	G	E
#	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
P	2	1	0	1	2	3	4
O	3	2	1	0	1	2	3
O	4	3	2	1	2	3	4
N	5	4	3	2	1	2	3

ФУНКЦИОНАЛЬНОЕ УРАВНЕНИЕ БЕЛЛМАНА

- Можем ли мы в задаче о расстоянии редактирования восстановить решение?
- Обратим внимание: может быть больше одного решения весом 3
- Также следует заметить, что у нас есть некоторый выбор: оптимизировать память или восстанавливать решение

	#	S	P	O	N	G	E
#	**0**	1	2	3	4	5	6
S	1	**0**	1	2	3	4	5
P	2	1	**0**	1	2	3	4
O	3	2	1	**0**	1	2	3
O	4	3	2	**1**	2	3	4
N	5	4	3	2	**1**	**2**	**3**

ВЕРНЕМСЯ К ПРЕДЕЛАМ РЕГУЛЯРНОСТИ

Мы хотели бы сматчить все правильные скобочные выражения и только их.

Для этого невозможно написать регулярное выражение, но можно написать грамматику

ГРАММАТИКА

По определению грамматика состоит из продукций $\alpha \to \beta$

$A \to A + A|(A)|A.A|A^*|a|b|c|\varepsilon$

Нетерминальные символы в общем случае могут стоять и слева и справа, **терминальные** – только справа

Для языка регулярных выражений на $\{a, b, c\}$

- Терминалы: $a, b, c, +, ., (,), *$
- Нетерминалы: пока только A

Обратим внимание: во всех продукциях языка регулярных выражений у нас слева всего один нетерминал

ПРИМЕР ПРОСТОЙ ГРАММАТИКИ

Контекстно-свободной грамматикой называется такая, которую можно представить так, чтобы слева в каждой продукции был ровно один нетерминал

Любой регулярный язык тривиально является контекстно-свободным

Язык $L_{parm} = \{a^m b^n c a^m b^n\}$ не является контекстно-свободным

Грамматика для скобочных выражений.

$B \to (B)\ B$

Возможные порождаемые скобки – любые скобочные выражения

`(()(()))`, `((()())())`

ПОРОЖДЕНИЕ

Порождаем все скобочные выражения

$B \to (B)\,B$

Сделаем левое порождение (( ) (( )))

$$B \to (B)B \to \big((B)B\big)B \to \big((\quad)B\big)B \to \big((\quad)(B)B\big)B \to \Big((\quad)\big((B)B\big)B\Big)\,B$$

Чтобы не порождать, а распознавать выражения, приведем грамматику в нормальную форму

НОРМАЛЬНАЯ ФОРМА ХОМСКОГО

Разрешены только следующего вида продукции

$$A \to a, A \to BC, S \to \varepsilon$$

Основная идея в замене $A \to aB$ на $A \to XB, X \to a$

Пример: $B \to (B)B|\varepsilon$

Как мы могли бы распознать строку (( ) (( )))?

$S \to LZ_1|LR|\varepsilon$
$B \to LZ_1|LR$
$Z_1 \to BZ_2|BR$
$Z_2 \to RB$
$R \to)$
$L \to ($

Благодаря свойствам нормальной формы, можно привести её в нетерминалы

L	L	R	L	L	R	R	R
(	(	)	(	(	)	)	)

ИДЕЯ ДЛЯ АЛГОРИТМА ПРИНАДЛЕЖНОСТИ

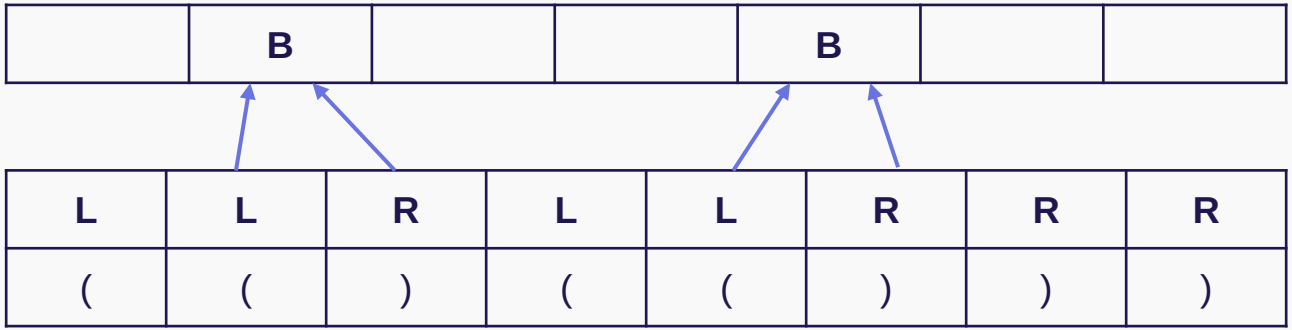
Далее попробуем комбинировать подстроки длины 2, длины 3 и так далее

Пусть мы ищем возможный вывод $S \to LLRLLRRR$

В этом случае мы должны рассматривать $X_i \to X_j X_k$

У нас три параметра: начало диапазона, длина и продукция

$S \to LZ_1|LR|\varepsilon$
$B \to LZ_1|LR$
$Z_1 \to BZ_2|BR$
$Z_2 \to RB$
$R \to)$
$L \to ($



ПРИНЦИП ОПТИМАЛЬНОСТИ

Для алгоритма CYK принцип оптимальности примет следующий вид

$$V(1, i, j) = true$$
$$V(k, i, a) = \max\big(V(p, i, b) \wedge V(k - p, i + p, c)\big), \text{где } X_a \to X_b X_c$$

Это сводится (снова) к заполнению двумерной таблицы. Но на этот раз мы должны рассматривать также диапазон индексов p от единицы до длины строки.

Что сразу приводит нас к простому псевдокоду

АЛГОРИТМ КОКА-ЯНГЕРА-КАСАМИ

Для строки с n символами и для k продукций, после начальной инициализации, алгоритм следующий:

```
for (l = 2, l < n; ++l)
  for (s = 1; s < n – l + 1; ++s)
    for (p = 1; p < l – 1; ++p)
      for each production Xa -> Xb Xc
        V[l, s, a] = max(V[l, s, a],
                         V[p, s, b], && V[l - p, s + p, c])
```

Реализуйте этот код на языке C++

Если в итоге `V[n, 1, 1] = true`, то слово принадлежит языку

ФОРМИРОВАНИЕ ИНТУИЦИИ

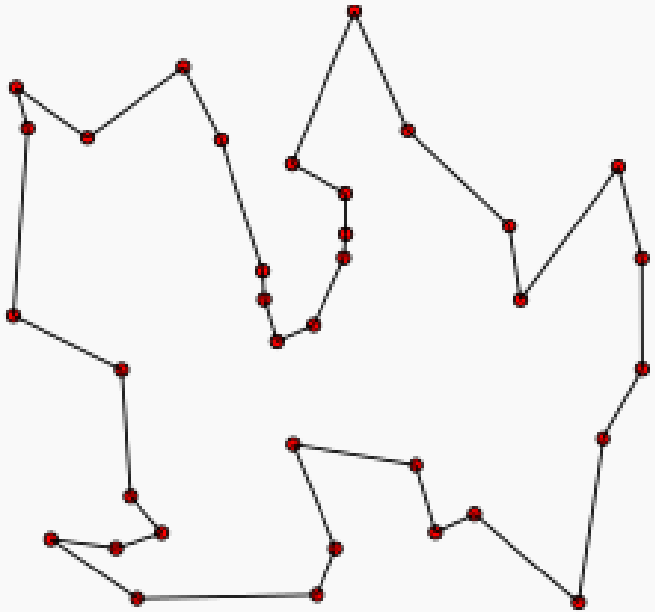
Не все задачи можно эффективно решить динамическим программированием

Пример: travelling salesman problem

Можно ли написать функциональное уравнение Беллмана? В общем можно

$$V(S,k) = \min\big(V(S-\{k\},m) + d(m,k)\big)$$

Увы, решение при этом будет $O(n^2 2^n)$ по времени и по памяти



УПРАЖНЕНИЯ

В кошельке лежат n монет. Номинал j-той монеты равен a_j. Требуется узнать, можно ли набрать с их помощью сумму k. Формально говоря, требуется проверить существует ли подмножество с заданной суммой.

Ограничения:

- Все числа натуральные
- $n \leq 100$
- $k \leq 10^6$
- $a_i \leq 10^4$

Запишите уравнение Беллмана и напишите программу, которая решает данную задачу

УПРАЖНЕНИЯ

В кошельке лежат n монет. Номинал j-той монеты равен a_j. Требуется определить наименьшее количество монет, чтобы с их помощью набрать сумму k. Если это невозможно, то функция должна вернуть -1.

Ограничения:

- Все числа натуральные
- $n \leq 100$
- $k \leq 10^6$
- $a_i \leq 10^4$

Запишите уравнение Беллмана и напишите программу, которая решает данную задачу

УПРАЖНЕНИЯ

Пусть дана матрица размером $n \times m$. Робот изначально находится в ячейке $(1,1)$. За один шаг робот может передвинуться на одну ячейку вправо или вниз. Требуется найти такой маршрут, сумма элементов которого максимальна.

Ограничения:

- Все числа целые и по модулю не превосходят 10^4
- $1 < m, n \leq 100$
- $k \leq 10^6$
- $a_i \leq 10^4$

Запишите уравнение Беллмана и напишите программу, которая выводит максимальную сумму

УПРАЖНЕНИЯ

Даны n целых чисел a_i и число $1 \le k \le n$. Требуется найти подотрезок длины k с наибольшим минимумом из всех возможных. Иными словам, требуется найти такую позицию p, которая максимизирует

$$\min_{p \le j < p+k} a_j$$

Ограничения:

- $n \le 5 \cdot 10^5$,
- $|a_i| \le 10^9$
- k и n - натуральные

Запишите уравнение Беллмана и напишите программу, которая выводит позицию начала подотрезка с наибольшим минимумом.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Бьерн Страуструп, Язык программирования C++/ ред. А. Боборыкин. – 4-е изд. - Москва: Издательство БИНОМ, 2023. – 1213 с.
2. Дональд Кнут Искусство программирования, том 1. Основные алгоритмы = The Art of Computer Programming, vol.1. Fundamental Algorithms. — 3-е изд. — М .: «Вильямс» , 2006. – 720 с.
3. Дональд Кнут Искусство программирования, том 2. Получисленные алгоритмы - The Art of Computer Programming, vol.2. Seminumerical Algorithms. — 3-е изд. — М .: «Вильямс» , 2007. — 832 с.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦМНО, 1999. – 960 с.
5. Скотт Мейерс, Эффективное использование C++. 55 верных способов улучшить структуру и код ваших программ / ред. Д.А. Мовчан – 3-е изд. – Москва: ДМК Пресс, 2017. – 300 с.
6. Скотт Мейерс, Наиболее эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов / ред. Д.А. Мовчан – 3-е изд. – Москва: ДМК Пресс, 2016. – 298 с.