ся k-я новая. Очевидно, начинается эта k-я ступенька на $(h_{k-1}+1)$-й старой. Добавочную площадь, показанную на рисунке 6, обозначим через $g(h_{k-1}+1,\ h_k)$. Общая задача состоит в том, чтобы найти значения h_1, $h_2, \ldots, h_{k-1}$, при которых сумма k величин

$$g(1, h_1)+g(h_1+1, h_2)+\ldots$$
$$\ldots+g(h_{k-2}+1, h_{k-1})+g(h_{k-1}+1, h_k)$$

(при заданном $k=n$ и $h_k=h$) наименьшая, обозначим это наименьшее значение через $f_n(h_n)$. Заметим, что

ке 3).

Обсуждение метода и немного истории

Обратим внимание на то, что в описанном методе не накладывается никаких условий на вид функций g (этим динамическое программирование выгодно отличается от линейного, в котором требуется, чтобы все рассматриваемые зависимости были линейными и переменные изменялись непрерывно). Существенно лишь, что функцию f, минимум которой мы

ищем, удается представить в виде суммы членов, в которой предыдущий зависит от меньшего числа переменных. Поэтому уравнение Р. Беллмана и вышеописанный метод используются сейчас очень широко. Но, разумеется, не для удовлетворения прихотей фараонов. Попробуйте без него найти оптимальное соотношение весов ступеней космической ракеты, *определить наилучший принцип унификации деталей*, построить расписание, в котором было бы меньше всего простоев! Вы очутитесь в отчаянном положении казначея, сникшего перед обилием возможных вариантов. Разумеется, вручную с помощью динамического программирования почти не считают. Зато оно хорошо приспособлено для счета на ЭВМ и не содержит всяких «подводных камней», которые, к сожалению, встречаются в линейном программировании и других методах отыскания минимумов для функций от многих переменных.

Идея метода динамического программирования витала уже давно. В какой-то мере ее применял еще К. Маклорен (1698—1746) и даже Архимед. Однако окончательно оформилась она в работах американского математика Р. Беллмана и связывает*ся с его именем.*

Мы здесь показали применение динамического программирования к весьма простой задаче. Насколько применим этот метод к задачам более

Еще одно приложение

Одно из сравнительно новых применений динамического программирования — молекулярная биология, где «банки данных» содержат биологические последовательности (ДНК, РНК, белков), насчитывающие в сумме многие миллионы «букв», так что в их анализе не обойтись без компьютеров. Молекулы ДНК, содержащие генетическую информацию,— это длинные слова из четырех букв (А, Г, Ц, Т). В процессе эволюции, в результате мутаций, последовательности меняются: одна буква может замениться на другую, выпасть, а может добавиться новая. Насколько похожи два фрагмента — каким наименьшим числом превращений можно один из них получить из другого? Вот точная формулировка задачи такого типа. *Заданы два «слова» (длины m и n), состоящие из букв А, Г, Ц, Т. Найти подпоследовательность наибольшей длины, входящую в то и другое слово.* Опишем простую, предложенную в начале 70-х годов, процедуру, которая решает эту задачу (биологи называют ее алгоритмом Нудельмана — Вунша). Запишем одно из данных слов — $x=(X_1,\ X_2,\ \ldots$ $\ldots,\ X_m)$ — снизу-вверх, а другое — $y=$

Ц	1	2	3	3	4	5	5	6	7
Т	1	2	2	3	4	5	5	6	7

сложным? На этот вопрос нужно отвечать с известной осторожностью. Дело в том, что при этом возникает специфическая трудность, которую назвали «проклятием размерности». Приходится перебирать так много разных возможностей и запоминать столько результатов, что метод теряет свои положительные черты. «Второе дыхание» метод динамического программирования обрел благодаря «распараллеливанию» вычислений, которому в последние годы уделяется все больше внимания специалистами по программированию и конструкторами вычислительных машин.

Г	1	2	2	3	4	4	5	6	6
Г	1	2	2	3	4	4	5	5	5
А	1	1	2	3	4	4	4	4	4
Т	1	1	2	3	3	4	4	4	4
А	1	1	2	3	3	3	3	3	3
Ц	0	1	2	2	2	2	2	2	2
Г	0	1	1	1	1	1	1	1	1
	А	Г	Ц	А	А	Т	Г	Г	Т

Рис. 7. Одна из последовательностей максимальной длины 7: ГЦААГГТ.

$$x = \_\;\boxed{\text{ГЦА}}\;\text{Т}\;\boxed{\text{А}}\;\_\;\boxed{\text{ГГТ}}\;\text{Ц}$$
$$y = \text{А}\;\boxed{\text{ГЦА}}\;\_\;\boxed{\text{А}}\;\text{Т}\;\boxed{\text{ГГТ}}\;\_$$

Рис. 8. Одно из наилучших «выравниваний» двух слов, найденное на рисунке 7.

$=(Y_1, Y_2, \ldots, Y_n)$ — слева-направо рядом с таблицей размерами $m \times n$ (рис. 7). Отметим те клетки таблицы, в строке и столбце каждой из которых стоят одинаковые буквы (в таблице на рисунке 7 художник закрасил эти клетки голубым цветом). Теперь будем заполнять таблицу числами a_{ij} $(i=1,2,\ldots,m;\ j=1,2,\ldots,n)$ по следующему правилу: a_{ij} равно наибольшему из чисел $a_{i-1,j}$, $a_{i,j-1}$ и (если клетка (i, j) отмечена!) $a_{i-1,j-1}+1$. Конечно, нужно задать еще начальные условия. Они таковы: $a_{11}=1$, если первые буквы X_1 и Y_1 наших слов совпадают (при этом $a_{ij}=a_{ji}=1$ для всех i, j); если же нет, то $a_{11}=a_{12}=\ldots=a_{1,j-1}=0$ и $a_{1j}=\ldots=a_{1n}=1$, если первые $j-1$ букв второго слова y отличны от X_1, а j-я буква $Y_j=X_1$; аналогично, $a_{21}=\ldots=a_{i-1,1}=0$, $a_{i1}=\ldots=a_{m1}=1$, если X_i — первая из букв слова x, совпадающая с Y_1.

Итак, вычисляя каждый раз максимум из двух или трех чисел,— и запоминая, как и в задаче фараона, наш путь,— мы за $m \cdot n$ шагов заполним всю таблицу.

букв свой «вес», отражающий степень их сходства (измеряемую, например, частотой замен одной буквы на другую). Очень важна и задача «множественного выравнивания» сразу нескольких слов, но с ростом числа слов количество необходимых операций в ней быстро растет.

Упражнения

1. Для школьной мастерской решили сначала купить 9 разных токарных станков стоимостью 10, 20, 40, 60, 75, 100, 130, 160, 190 рублей. Каждый из последующих станков в этом ряду может заменить любой предыдущий, но не наоборот. Например, станок за 100 рублей может обрабатывать те же детали, которые обрабатывает станок за 40 рублей. Однако директор школы возразил:

— Слишком много типов станков! Нам будет трудно их обслуживать — ведь для каждого нужны свои запчасти. Давайте купим 9 станков, но всего четырех разных типов, причем так, чтобы купленные станки обладали не меньшими возможностями, чем исходные 9 станков.

— А какие типы станков мы возьмем?

— Выберем их так, чтобы заплатить за все 9 станков поменьше. Евгений Иванович, вы поможете нам выбрать типы станков? — обратился он к учителю математики.

Какие типы станков предложил выбрать учитель и сколько они стоили?

2. То, что предложил проделать директор в предыдущей задаче, называется унификацией. Обычно на практике унификация дает экономическую выгоду. Но как эту выгоду подсчитать?

Пусть снова имеется 9 станков со стоимостями 16, 18, 19, 22, 23, 24, 26, 28, 33 рубля. Требуется составить из них унифицированные серии, как это было сделано раньше, считая, что объединение станков в серии за счет упрощения обслуживания дает снижение затрат на серию по следующей таблице:

Проверьте, что последнее число a_{mn} будет равно длине наибольшей общей подпоследовательности наших двух слов, а саму эту подпоследовательность можно прочесть, двигаясь в обратном порядке (иногда, возможно, путь и не единственный). Биологи обычно изображают результат как «выравнивание» (*alignement*) двух данных слов (рис. 8). Сходные быстрые и требующие мало памяти алгоритмы можно найти и для других похожих задач: например, замене букв можно приписать другой «вес», чем удалению (или вставке); в белковых последовательностях (это — слова из 20 аминокислотных «букв») разумно сопоставить каждой паре

Число станков в серии	1	2	3	4	5	6	7	8	9
Снижение стоимости	0	12	16	18	20	20	21	21	21

3. Дополните таблицы на рисунке 2, чтобы найти самую экономную лестницу из 5 ступеней; сколько золота на нее потребуется?