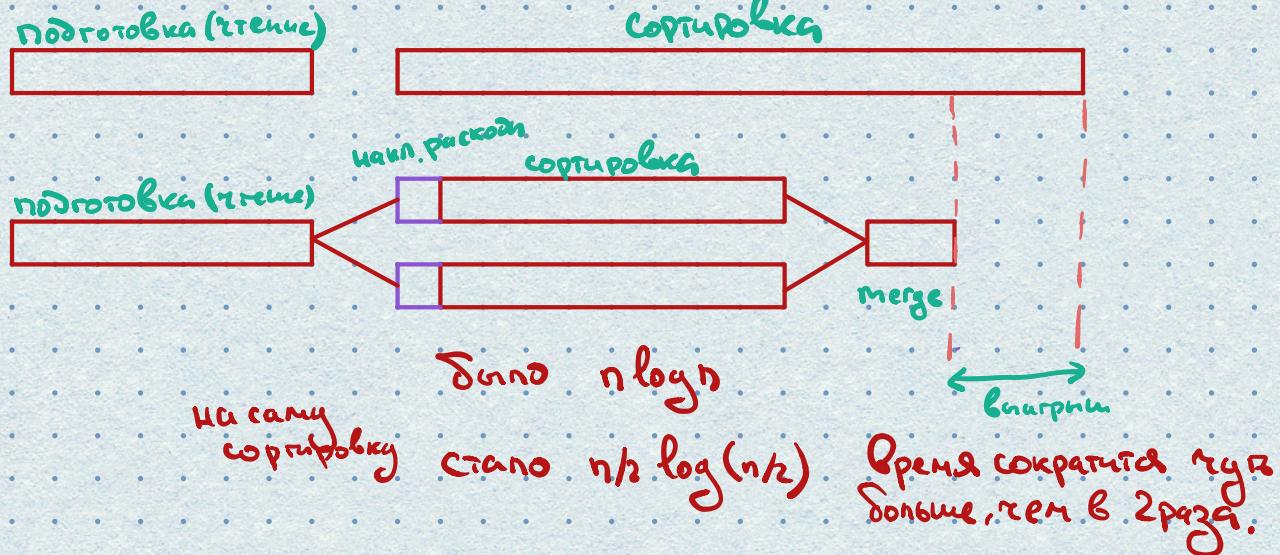


Сортировка



Выигрыш может быть отрицательным, если данных мало, то начальные расходы на merge и параллельность могут быть больше, чем выигрыш.

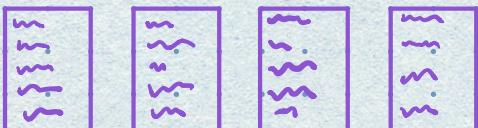
| Если есть часть, которую нельзя распараллить и она занимает половину времени, то выигрыш больше 2x раз всегда получим

Если будет больше, чем на 2 части, то можно merge запускать тоже параллельно.

Map-Reduce

Map - принимает данные и функцию, которую нужно применить.
Reduce - склоняет результат всех map

Пример:



посчитать кол-во слов в файлах

МНР: 15 36 8 94 независимо считает в каждом фрейм

Reduce: $15 + 36 + 8 + 94$ суммирует все результаты

Предфиксная сумма

3 1 2 4 6 1 хотят исключить MR
3 4 6 10 16 12

Иdea 1:

Поделим
пополам
считаем для
каждой половины
отдельно ипотом
прибавлен последний
элемент 1-го потока
ко всем эл. 2-го пот.

3	4	6	4	10	11
			+6	+6	+6

последоват. Время

суммы
каждой
половины

{

прибавлен эл.

паралл. Время

Всегда одна пог.

Иdea 2.

1. Делю на 4 части
2. находим пр. сумму для каждого: S_1, S_2, S_3, S_4
3. Для S находим пр. суммы: $S_1, S_1+S_2, S_1+S_2+S_3, \dots$
4. можем прибавлять эл. к массиву независимо

T

$|S_1| |S_2| |S_3| |S_4|$

$|S_1| |S_2| |S_3| |S_4|$

$|S_1| |S_2| |S_3| |S_4|$

T

$$P_1 = S_1; P_2 = S_1 + S_2, P_3 = S_1 + S_2 + S_3, \dots$$

$|S_1| \xrightarrow{+P_1} |S_2| \xrightarrow{+P_2} |S_3| \xrightarrow{+P_3} |S_4|$

могут распараллелить
суммирование.

$$\frac{1}{4}T + 4 + \frac{1}{4}T = \frac{1}{2}T + 4$$

предфикс.
суммы

P
читаем прибавлен P.

Но 4 потока мало.

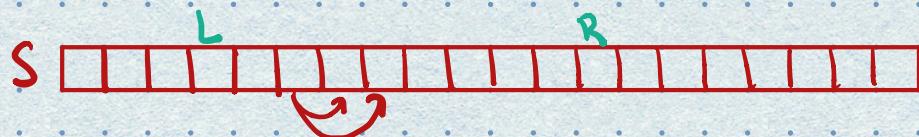
модиф для большего кол-ва

Если боязне нольков, то 4 может стать
относительно боязням спасением.

Тогда подает Р можно тоже распараллелить. Повторяя рекурсию

позвитя лог в сложности.

Рекурсивное программирование:



$$d[i] = S[i] + \max(d[i-1], d[i-2])$$

нужно набрать max сумму.
можно идти на 1 или 2.

Значение $d[i]$ зависит от предыдущих \rightarrow так проще не распараллелить.

Решение: динамику по префиксу заменить
на динамику по подотрезку.

Хотим наилучшим образом пройти весь от L до R

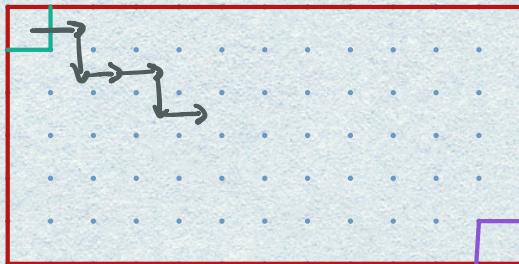
$$d[L, R] = d[L, M] + d[M+1, R]$$

$$M = (L+R)/2$$

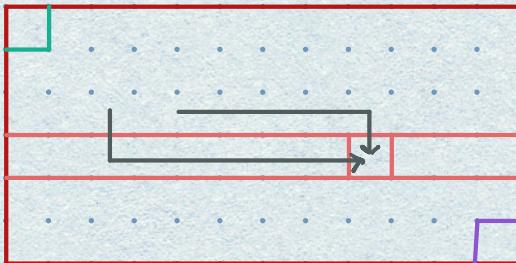
Пропущен конец!

Задача переноски.

Меренажи может идти на 1 вправо или на 1 влево сдвигая пасеку
надо набрать макс. количество



$$d[i, j] = c_{i,j} + \max(d[i-1, j], d[i, j-1])$$



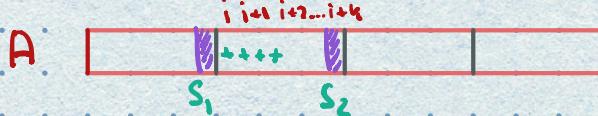
читает ~~эту~~ строку, путь все время уже посчитан.



помещен в ~~две~~ строку или сверху или внизу сколько-то ячеек слева.



разбить строку на части и будем асинхронно запускать.



для каждого блока рассмотрим все варианты поместив сверху, но упускаем вариант, если пропущены слева.

Но! Значит, что в предыдущем блоке в последнем блоке хранился стоимость лучшего пути. Можно для нее посчитать преф. сумму в начале блока и выбрать лучше.

Выбираем между S_2 и

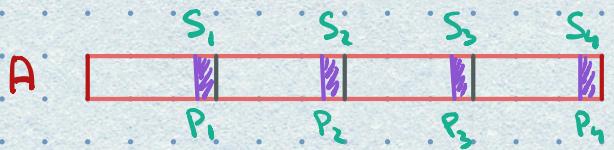
$$S_1 + A[i] + A[i+1] + \dots + A[i+k]$$

(если блок горизонт.)

$$T/4 + T/4 + T/4 + T/4$$

посчитать
в каждом
блоке лучше
маркируя сверху

пересчитать S_4 ,
пересчитать S_2 ,
пересчитать S_3



1. Тратим $T/4$, чтобы посчитать лучше варианты для блоков

$$P_1 = S_1, P_2 = \max(S_2, P_1 + \text{pr.sum}(S_2)).$$

2. Параллельно посчитали преф. сумму на отрезках в блоках.

3. Пересчитаем для ~~и~~ лучше значение.

или иначе сверху или
чрезр. сутки + конк. предодолев. блоки.

4. Пересчитываем все прошлогодние
запасы параллельно.
Мы можем так сделать, т.к.
запасы все Pi

$$T/4 + 4 + T/4$$


нечетные ячейки промежутка.

считаем	считаем	нечетные ячейки промежутка.
S_i	P_i	

+ если много здер, то пред. сущим
читаем Во Время первого прохода.
тогда субъективное Время не изменится.