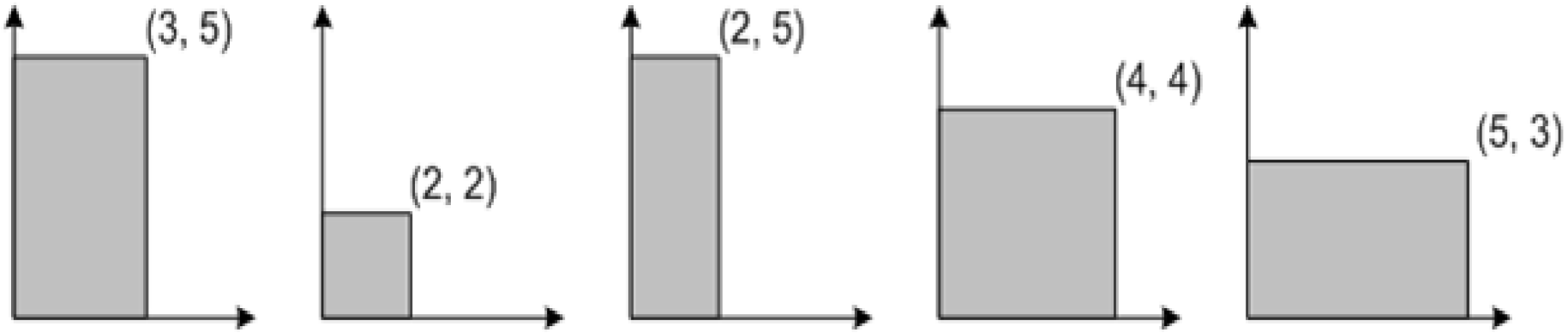
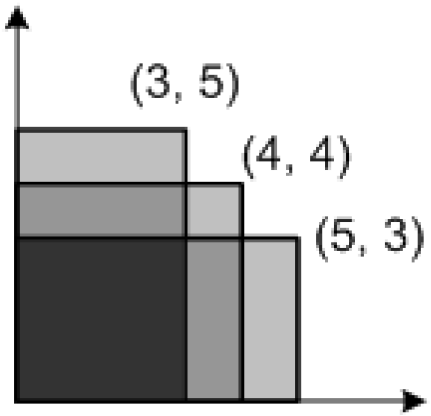
Задача 7: Силовые поля

Задача ориентирована на более подготовленных участников.
Основными темами задачи 7 «Силовые поля» являются структуры данных, сканирующая прямая.
Возможные частичные решения основаны на использовании перебора, сортировки.



$n = 5, \quad k = 3$



Отсортируем все прямоугольники по убыванию x_i.
Рассмотрим первые k штук прямоугольников.
Поместим их в структуру данных Q, позволяющую добавлять элементы, получать и удалять минимальный по значению высоты поля y_j элемент.

$x_n \leq \ldots \leq x_{k+1} \leq x_k \quad \ldots \quad x_2 \leq x_1$

$y_n \quad \ldots \quad y_{k+1} \quad y_k \quad \ldots \quad y_2 \quad y_1$

В качестве структуры Q можно использовать, например, дерево отрезков.

Вычислим площадь пересечения этих прямоугольников как произведение текущего значения ширины x_k и минимального значения y_{min}.

$$\text{Площадь пересечения } S = x_k * y_{min}$$

Затем перейдем к следующему прямоугольнику. Рассмотрим его высоту y_{k+1}. Если оно меньше минимального значения в Q, то нет смысла брать этот прямоугольник, его использование будет заведомо хуже уже рассмотренных вариантов. Иначе удалим из Q прямоугольник с минимальным y_{min}, добавим в него наш прямоугольник и обновим площадь пересечения прямоугольников, вычислив произведение x_{k+1} и нового минимального значения y_{min}. Рассмотрев таким образом все прямоугольники, получим оптимальный ответ.

```
program power;
const
                MAX = 200000;
type
 trect = record
   x, y: int64;
   py: longint; — будет нумеровать прямоугольники по убыванию их высоты
 end;
tarr = array [1..MAX] of trect;
...
var
   p: tarr;
...
for i := 1 to n do begin
read(p[i].x, p[i].y);
end;
```

Процедура сортировки прямоугольников по убыванию ширины x_i

```
procedure sortx(var p: tarr; l, r: longint);
var
    i, j: longint;
    supp, tmp: trect;
begin
    supp := p[l + random(r - l + 1)];
    i := l;
    j := r;
    while i <= j do begin
        while (p[i].x > supp.x) do
            inc(i);
        while (p[j].x < supp.x) do
            dec(j);
        if i <= j then begin
            tmp := p[i]; p[i] := p[j]; p[j] := tmp;
            inc(i);
            dec(j);
        end;
    end;
    if l < j then
        sortx(p, l, j);
    if i < r then
        sortx(p, i, r);
end;
```

```
sorty(p, 1, n);
for i := 1 to n do begin
    p[i].py := i;
    ys[i] := p[i].y;
end;

sortx(p, 1, n);
ans := 0;
for i := 1 to n do begin
    update(1, 1, n, p[i].py);
    if i >= k then begin
        y := ys[findk(1, 1, n, k)];
        if p[i].x * y > ans then
            ans := p[i].x * y;
    end;
end;
```

```
var
        s: array [1..4 * MAX] of longint;

procedure update(p, l, r, d: longint);
p – номер текущей вершины дерева отрезков
l, r – левая и правая границы отрезка , соответствующего текущей вершины
d – индекс меняющегося элемента
var
        m: longint;
begin
        if l = r then begin
            s[p] := s[p] + 1;
            exit;
        end;
        m := (l + r) div 2;
        if d <= m then
            update(2 * p, l, m, d)
        else
            update(2 * p + 1, m + 1, r, d);
        s[p] := s[2 * p] + s[2 * p + 1];
end;
```

Функция, находящая номер прямоугольника с минимальной высотой в ряду по убыванию высот

```
function findk(p, l, r, v: longint): longint;
var
        m: longint;
begin
        if l = r then begin
            findk := l;
            exit;
        end;
        m := (l + r) div 2;
        if s[2 * p] >= v then
            findk := findk(2 * p, l, m, v)
        else                findk := findk(2 * p + 1, m + 1, r, v - s[2 * p]);
end;
```

Оценка на время работы такого решения: $O(n \log n)$

Подзадача	Баллы	n	k	Необходимые подзадачи
1	18	$1 \le n \le 20$	$1 \le k \le n$	
2	25	$1 \le n \le 300$	$1 \le k \le n$	1
3	20	$1 \le n \le 3\,000$	$1 \le k \le n$	1, 2
4	17	$2 \le n \le 200\,000$	$k = 2$	
5	20	$1 \le n \le 200\,000$	$1 \le k \le n$	1, 2, 3, 4