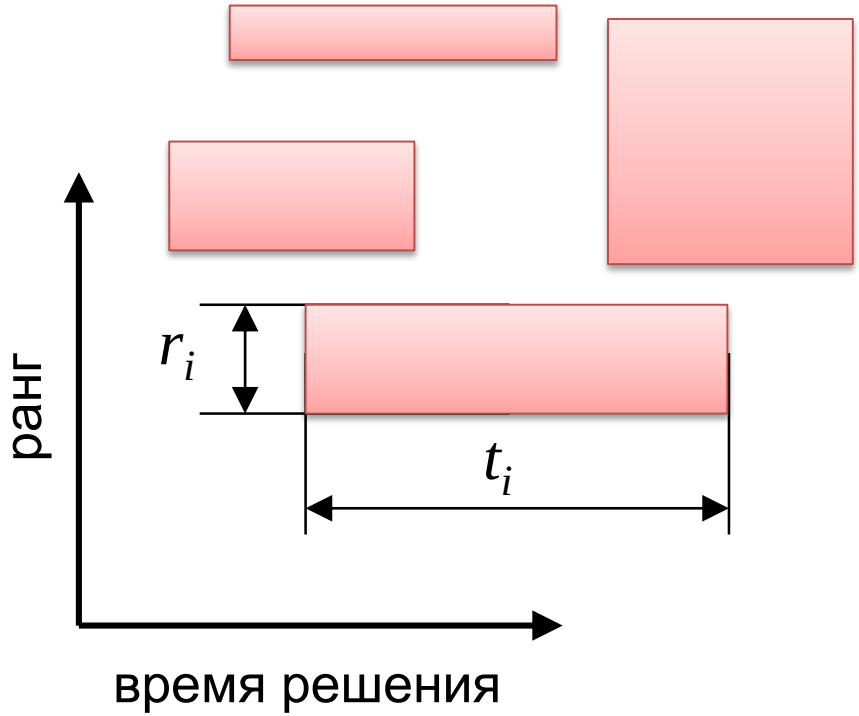
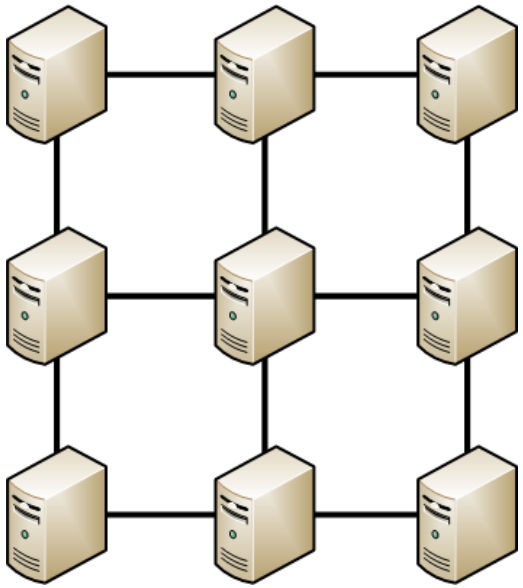
Обработка набора задач





Распределённая ВС

Постановка задачи

$J = \{1, 2, \ldots, m\}$ – множество задач

r_j – ранг параллельной задачи,

t_j – время решения;

$C = \{1, 2, \ldots, n\}$ – множество элементарных машин (ЭМ)

$S = (\tau_1, \tau_1, \ldots, \tau_m; x_{11}, x_{12}, \ldots, x_{1r_1}; x_{21}, x_{22}, \ldots, x_{2r_2}, \ldots, x_{mr_m})$

τ_j – время начала решения задачи j,

Ω – множество допустимых расписаний.

$$J(t) = \{j \in J \mid \tau_j \le t \le \tau_j + t_j\}$$

$$J'(t) = \{(j, i) \mid j \in J, 1 \le i \le r_j\}$$

$$C_{\max}(S) = \max_{j \in J}\{\tau_j + t_j\} \to \min_{S \in \Omega}$$

Постановка задачи

$$C_{\max}(S) = \max_{j \in J}\{\tau_j + t_j\} \to \min_{S \in \Omega}$$

при ограничениях:

$$\sum_{j \in J(t)} r_j \le n, \quad \forall t \in \Re,$$

$$\prod_{j \in J(t)} \prod_{j' \in J(t) \setminus \{j\}} (x_{ji} - x_{j'i'}) \neq 0, \quad \forall t \in \Re, \quad i = 1,2,\ldots,r_j, i' = 1,2,\ldots,r_{j'}$$

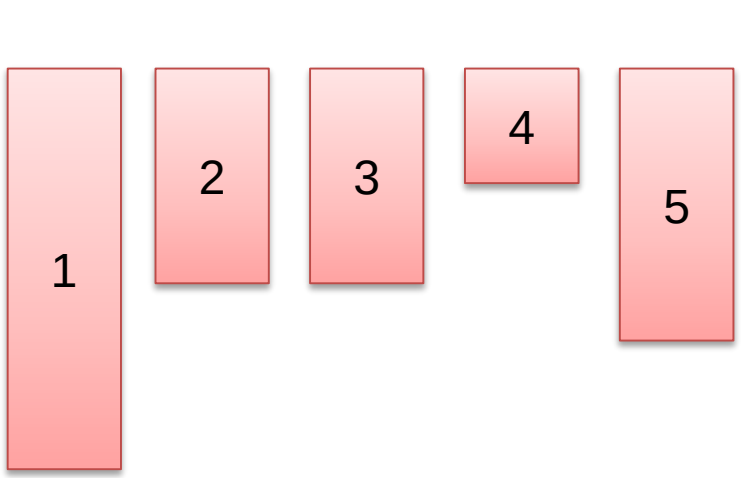
$$\tau_j \in \Re,\ j = 1,2,\ldots,m.$$

$$x_{ji} \in C,\ j = 1,2,\ldots,m,\ i = 1,2,\ldots,r_j$$

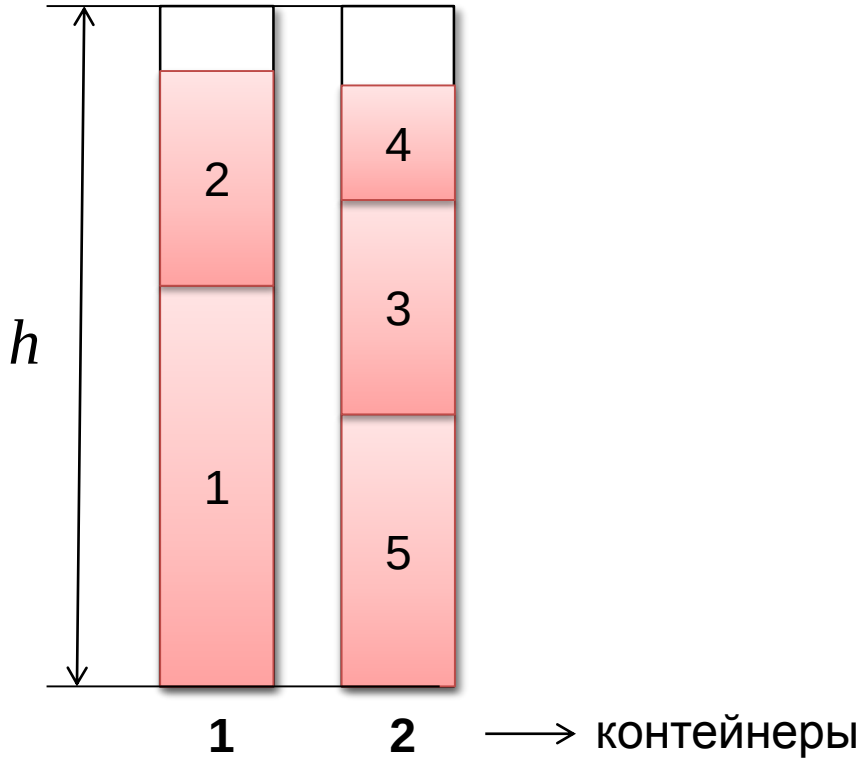
Алгоритм 1D Bin Packing

Имеется n предметов высоты h_i, $i \in \{1, 2, \ldots, n\}$

Требуется упаковать предметы в минимальное количество контейнеров высоты h



объекты



Эта задача трудноразрешима (*NP*-Complete)

NP-полнота

• Означает, что оптимальная упаковка в худшем случае не может быть получена за полиномиальное время (иначе, $P = NP$)

• Если будет найдено оптимальное решение какой-либо задачи за полиномиальное время, то все другие *NP*-полные задачи могут быть также решены за полиномиальное время.

- Задача о раскраске графа
- Задача о клике
- Задача коммивояжёра
- Задача о вершинном покрытии
- и т.д.

• См. М. Гэри, Д. Джонсон: Вычислительные машины и труднорешаемые задачи

NP-полнота

"Я не могу найти эффективного алгоритма, но этого не может сделать и никто из этих знаменитостей".

Эвристический алгоритм FFD

Вычислительная сложность FFD: $O(n\log_2 n)$

Задача имеет два параметра (r, t). Необходимо избавиться от одного параметра.

Будем строить “виртуальные задачи” – пакеты задач.

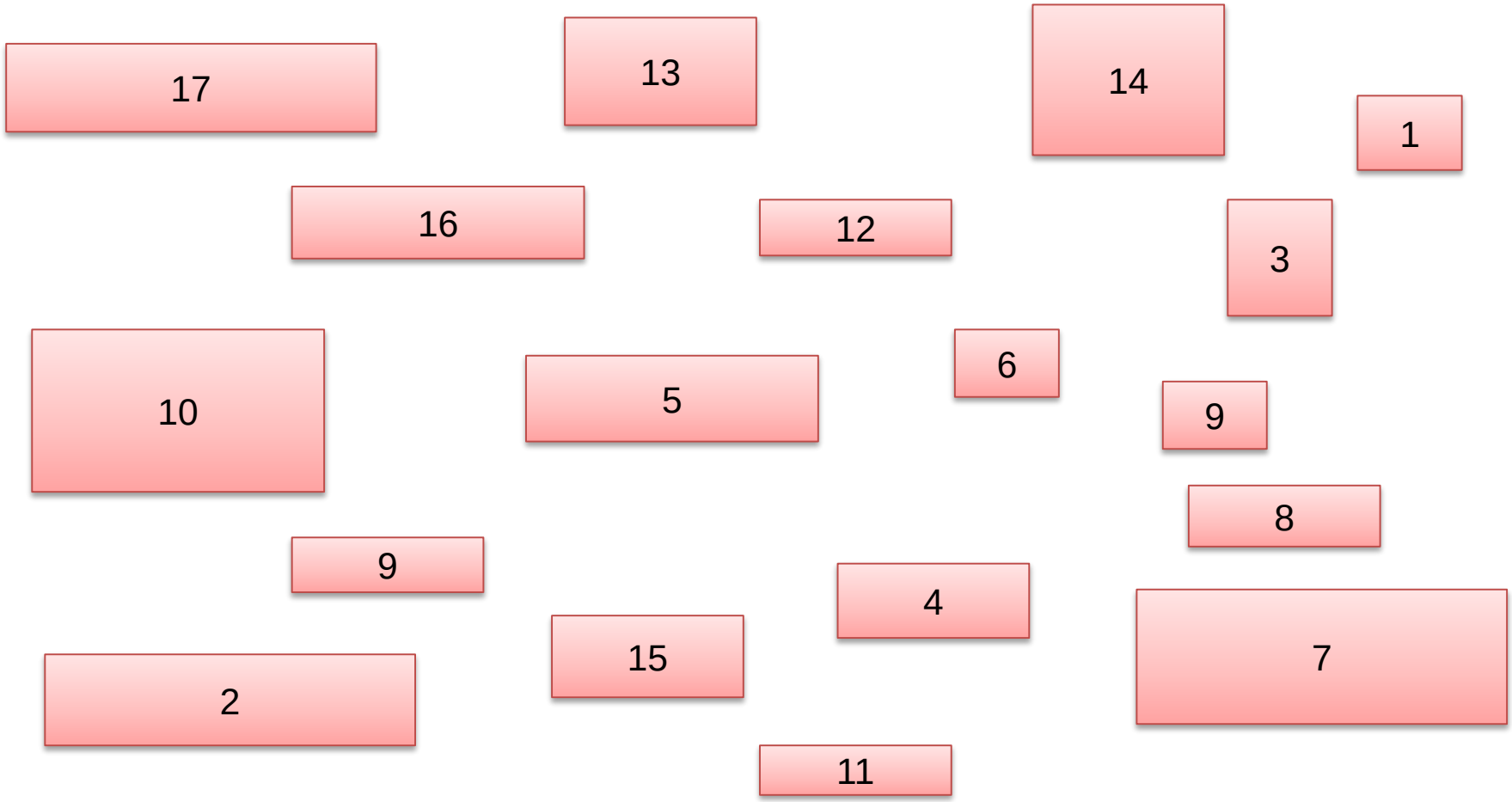
1. Выбираем все задачи ранга r $(O(m))$

   – Сортировка $O(m)$

   – Построение укрупнённых задач ранга r, у которых время решения близко к Θ, $\Theta >> t_j$

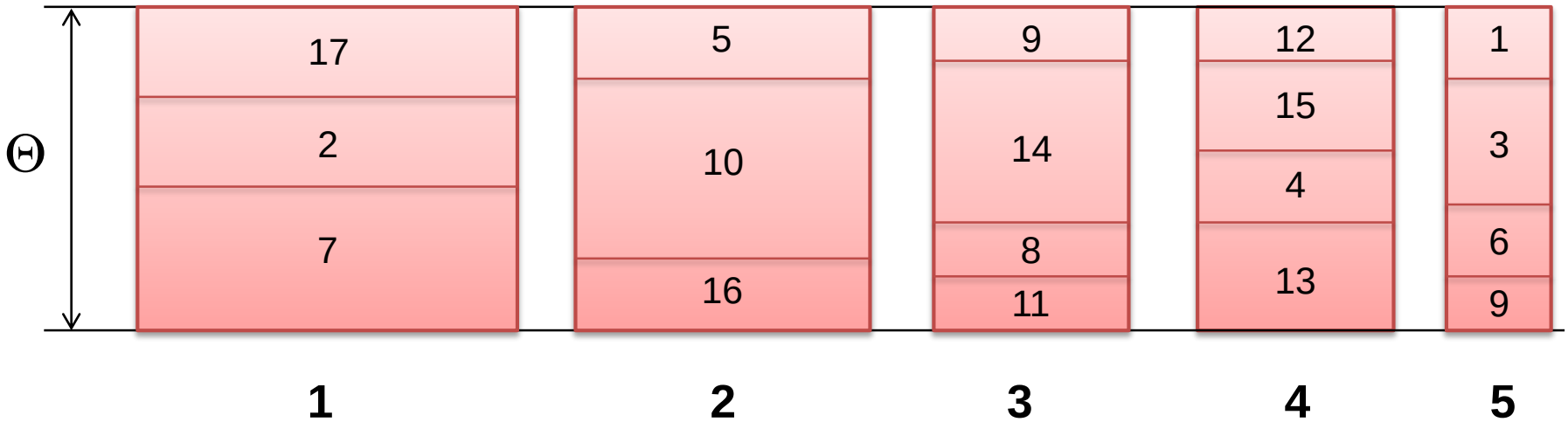
Эвристический алгоритм FFD

Имеется m задач различных рангов



Эвристический алгоритм FFD

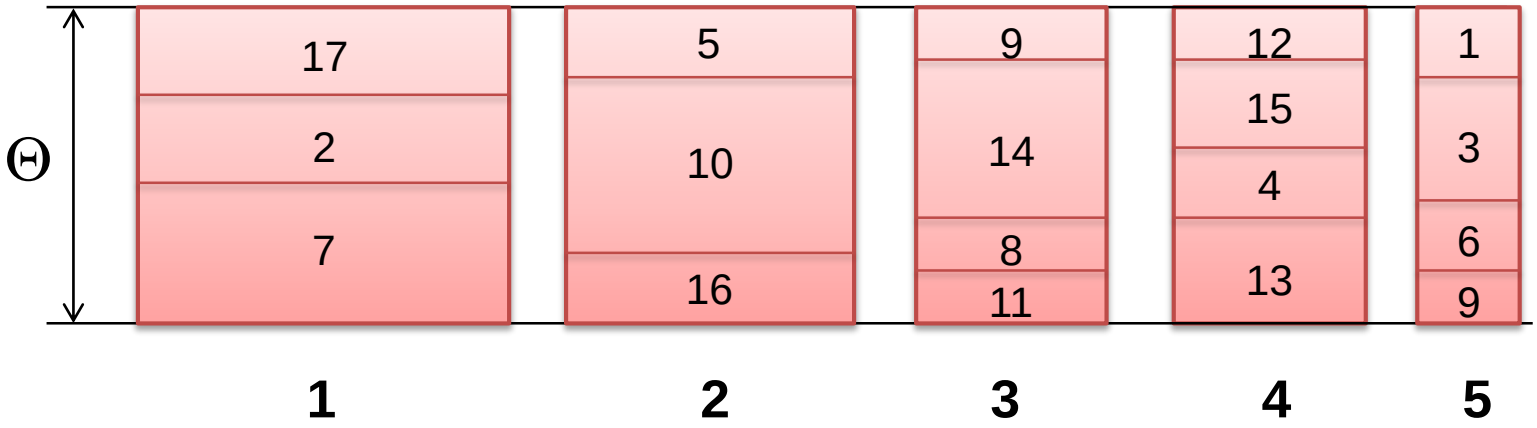
1. Формируем m' укрупнённых задач ранга r_j и времени решения Θ



$$t_1 = t_2 = t_3 = t_4 = t_5 = \Theta$$

$$r_1 = 4, r_2 = 3, r_3 = r_4 = 2, r_5 = 1$$

Эвристический алгоритм FFD



$$t_1 = t_2 = t_3 = t_4 = t_5 = \Theta$$

$$r_1 = 4,\ r_2 = 3,\ r_3 = r_4 = 2,\ r_5 = 1$$

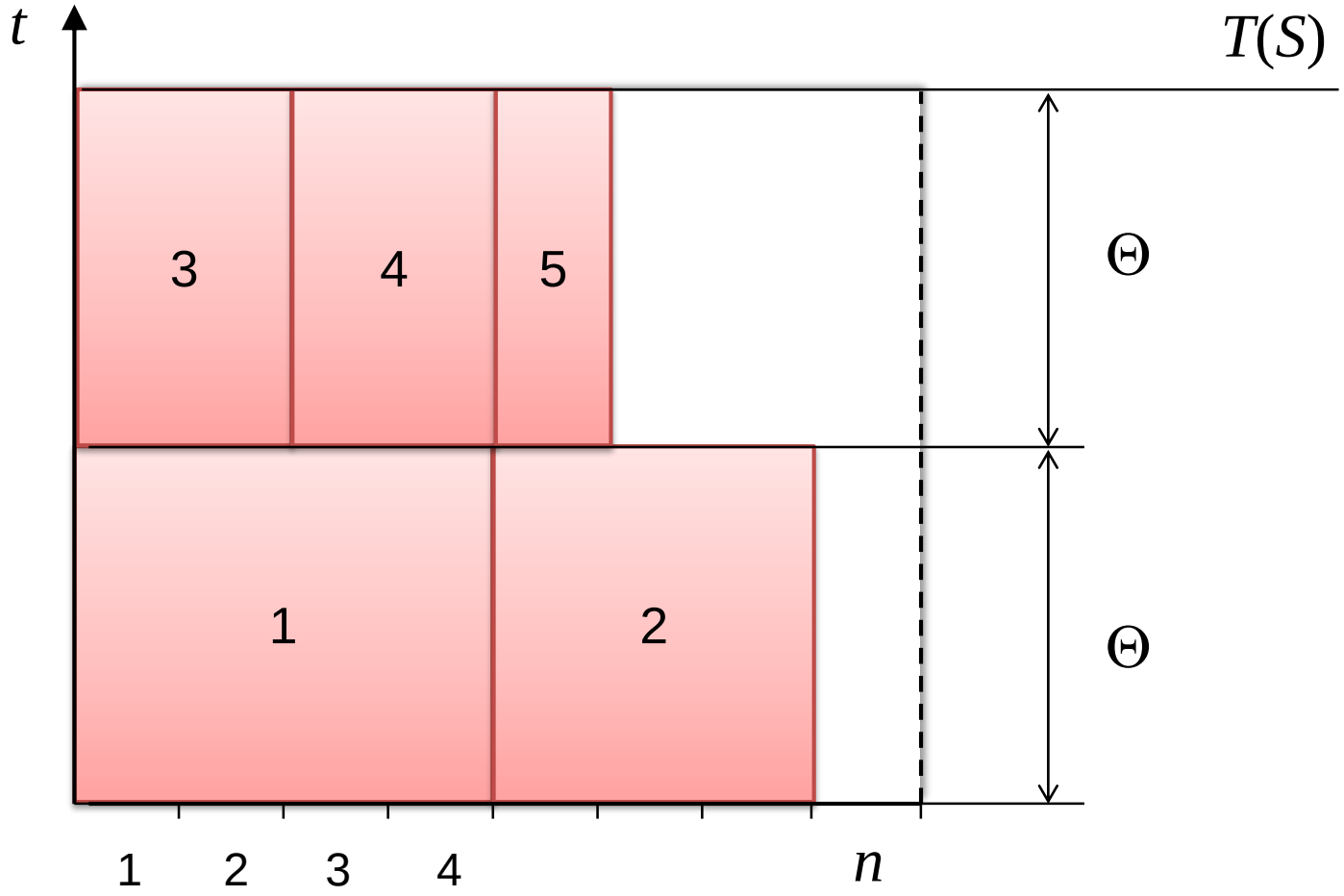
Как выбирать Θ? $\Theta = 10 \cdot \max\{t_i\}$

Как заполнять пакеты?

- **First Fit (FF)**: сортируем задачи в порядке неубывания, помещаем в пакет *первую подходящую* по размеру задачу.
- **Best Fit (BF)**: помещаем в каждый пакет задачу, которая обеспечивает *наименьшее оставшееся пространство* в пакете.

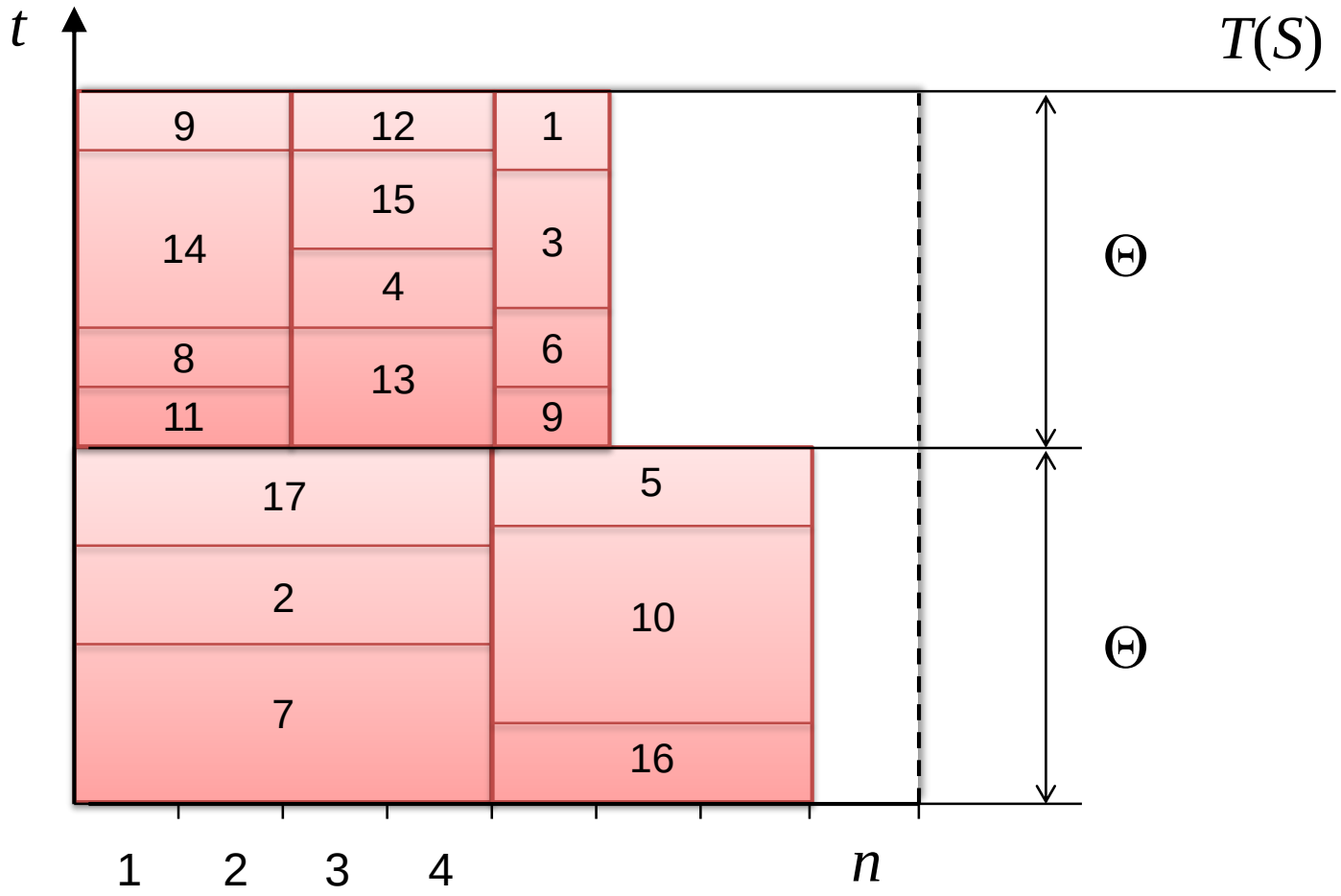
Эвристический алгоритм FFD

2. Распределение с помощью FFD укрупнённых задач



Эвристический алгоритм FFD

3. Восстанавливаем расписание



Эвристический алгоритм FFD

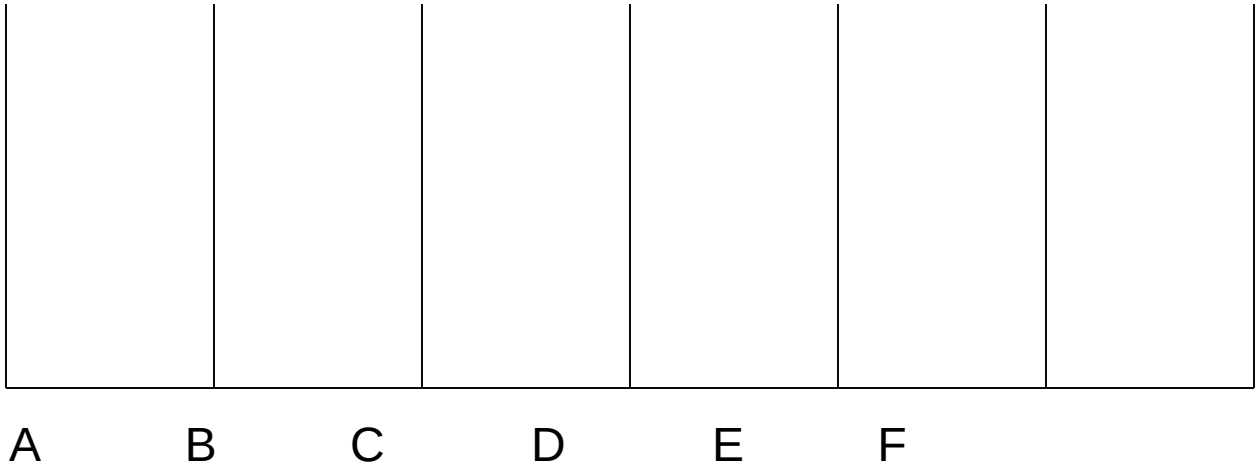
1
```
for i =1 to m do
    BuildPacks(i);    // множество укрупнённых задач
end for
```

2
```
for i = 1 to m' do
    BuildPacks(i);    // построение расписания
end for
```

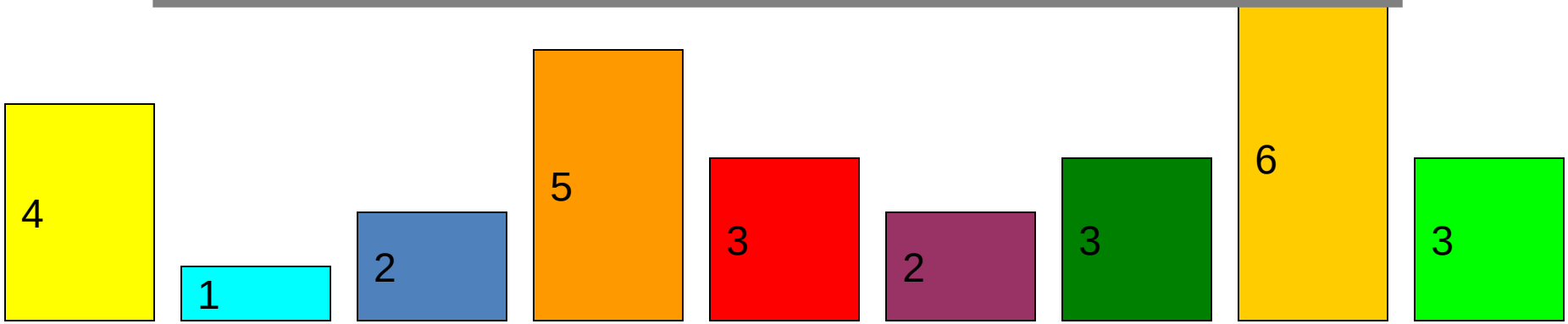
Трудоёмкость:

$$O(m\log_2 m + m'\log_2 m') = O(m\log_2 m)$$

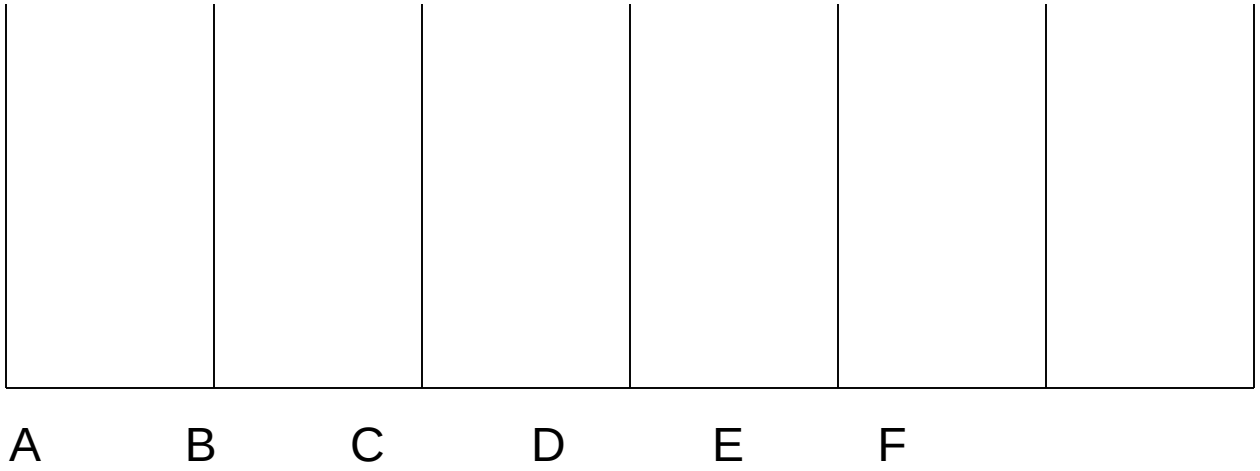
Bin Packing
First fit decreasing algorithm



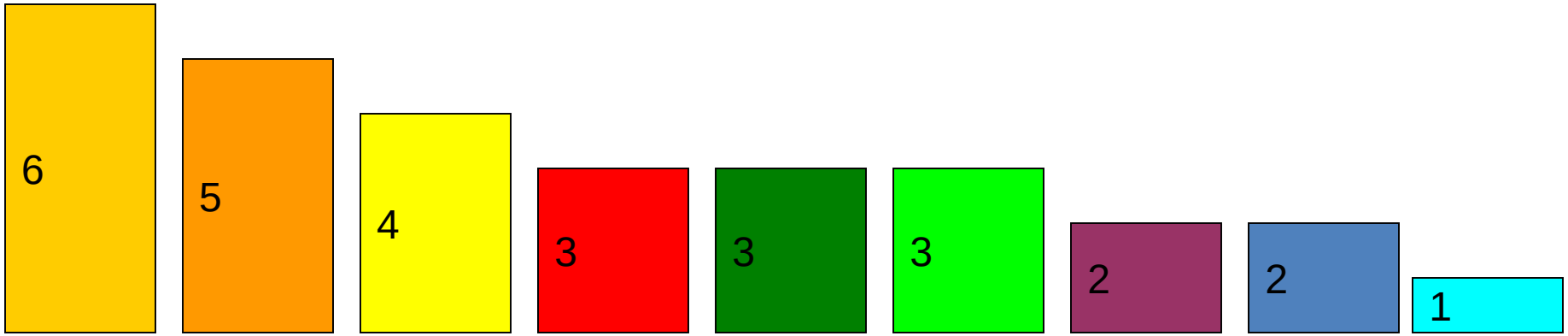
With the first fit decreasing algorithm we sort the blocks into descending order first.



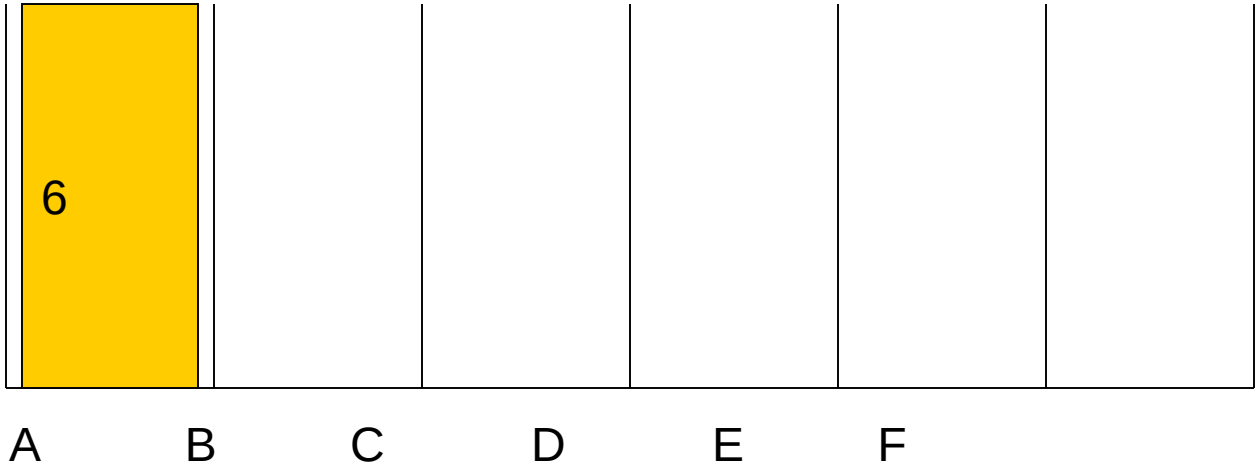
Bin Packing
First fit decreasing algorithm



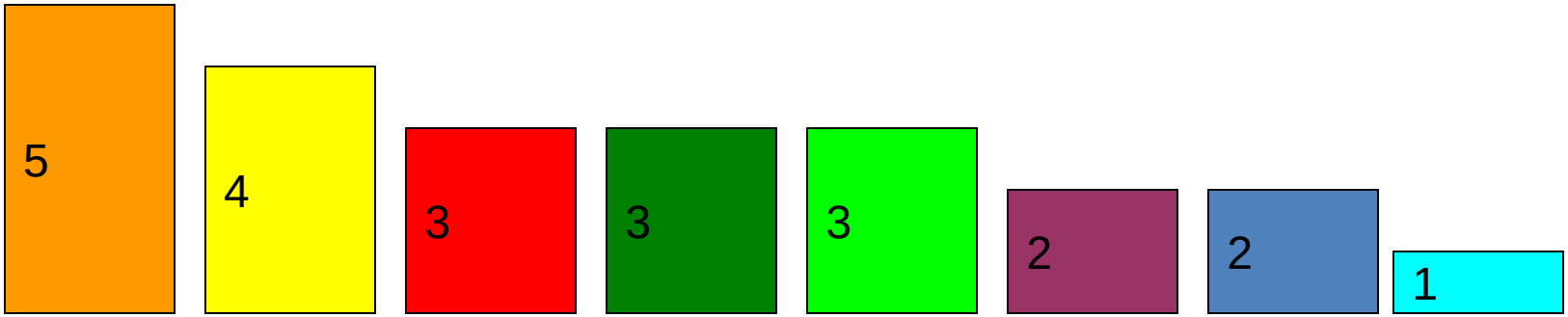
Now we use the first fit algorithm



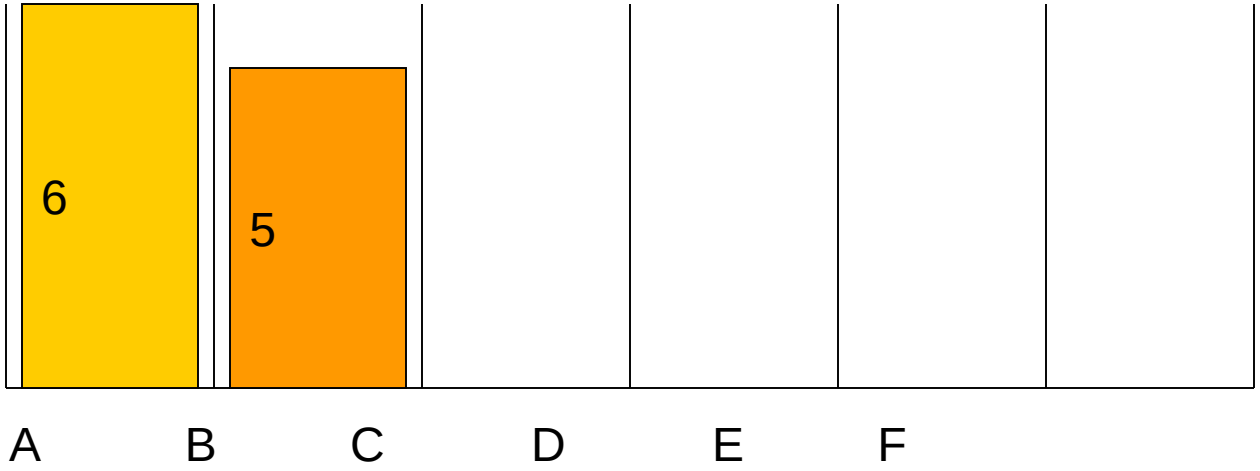
Bin Packing
First fit decreasing algorithm



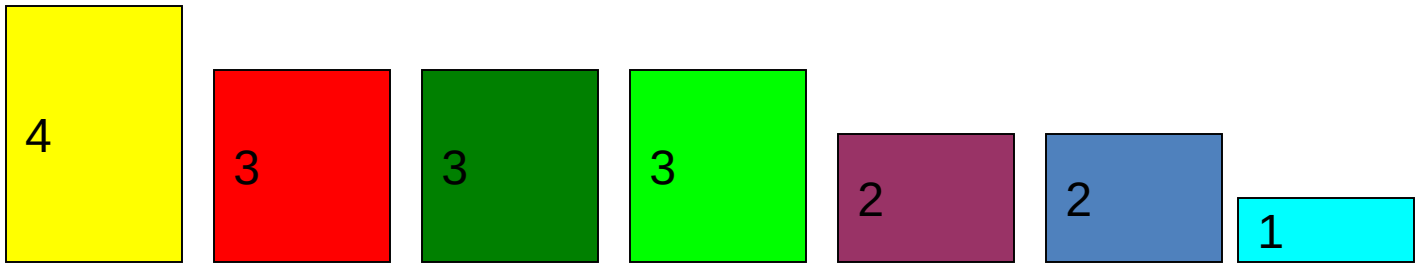
Now we use the first fit algorithm



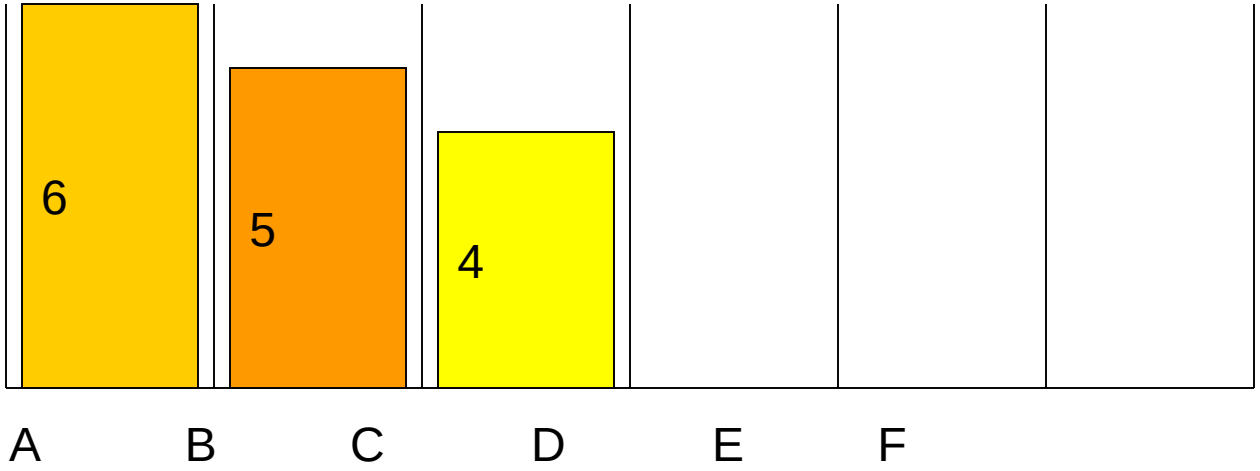
Bin Packing
First fit decreasing algorithm



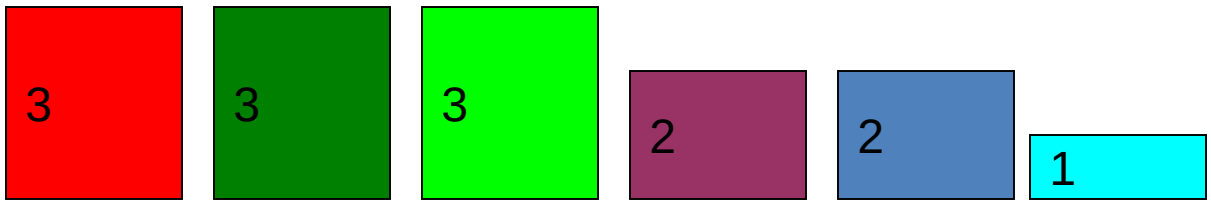
Now we use the first fit algorithm



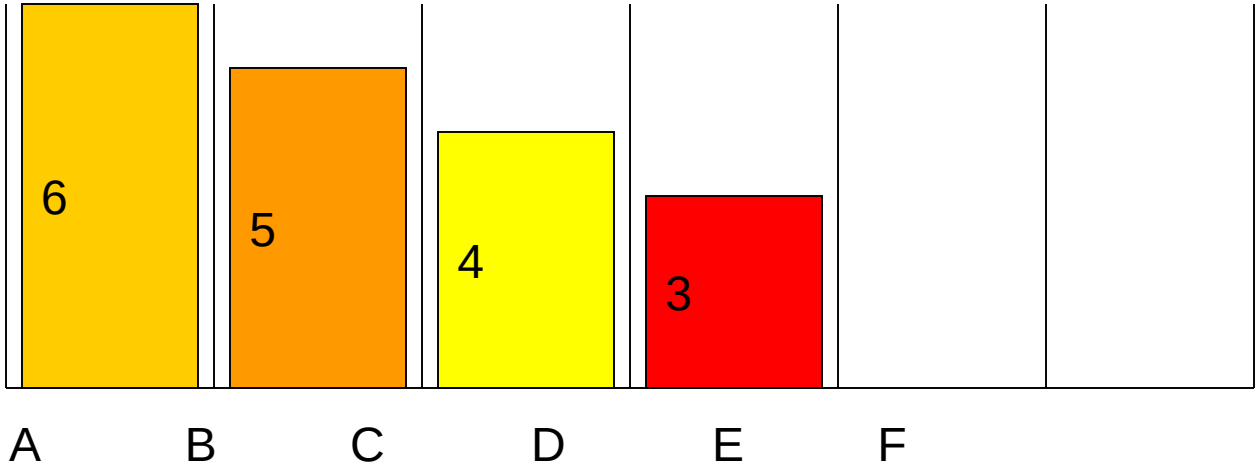
Bin Packing
First fit decreasing algorithm



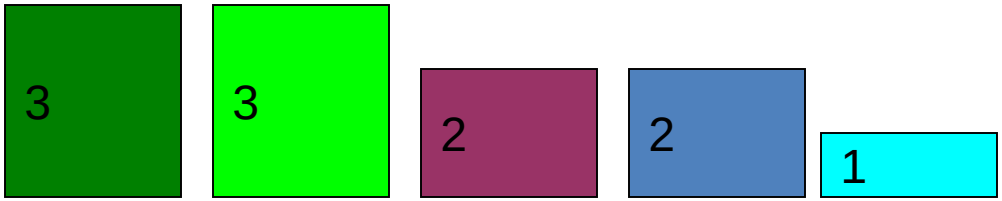
Now we use the first fit algorithm



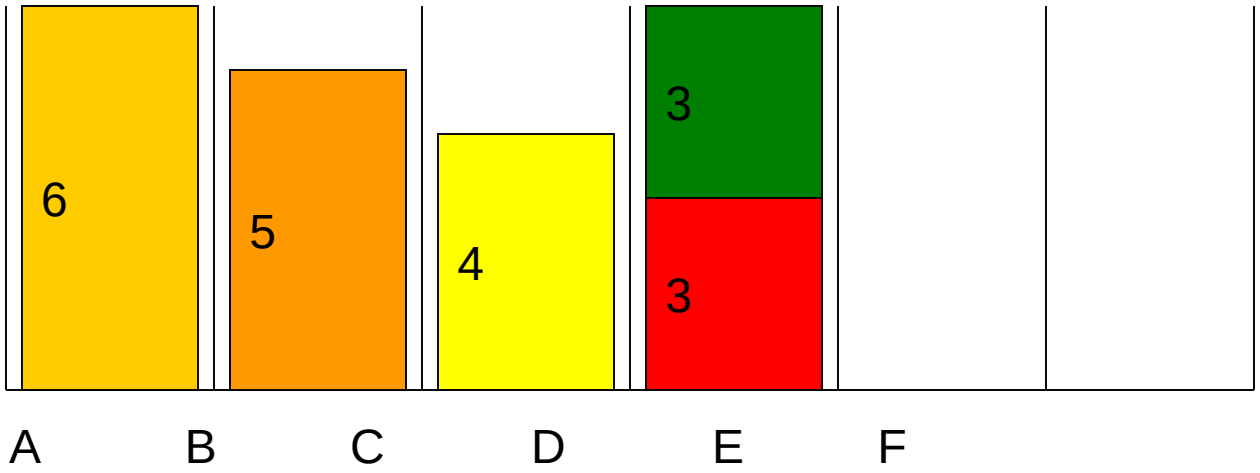
Bin Packing
First fit decreasing algorithm



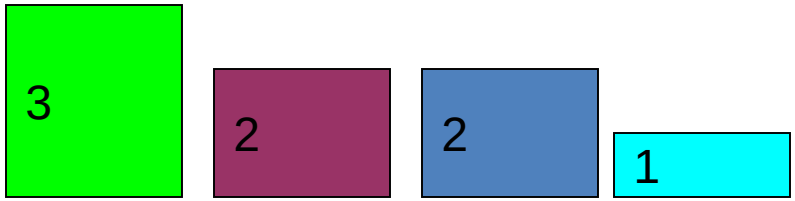
Now we use the first fit algorithm



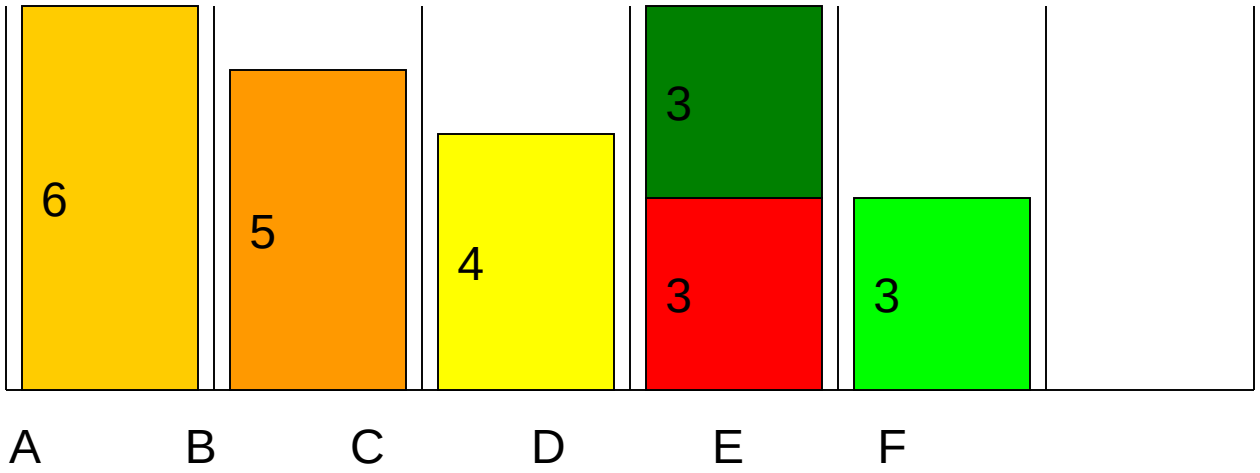
Bin Packing
First fit decreasing algorithm



Now we use the first fit algorithm

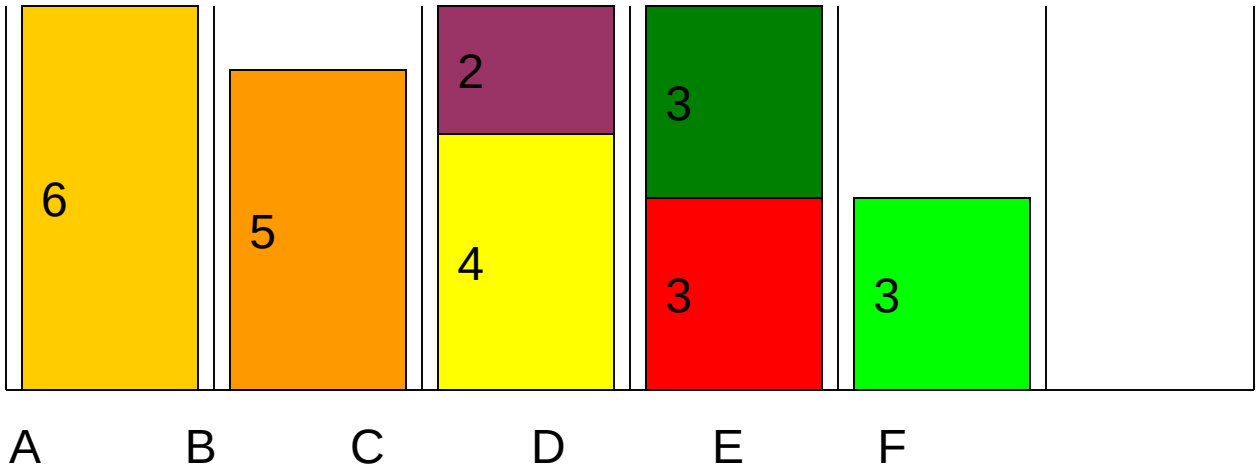


Bin Packing
First fit decreasing algorithm

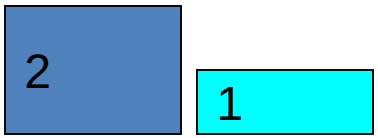


Now we use the first fit algorithm

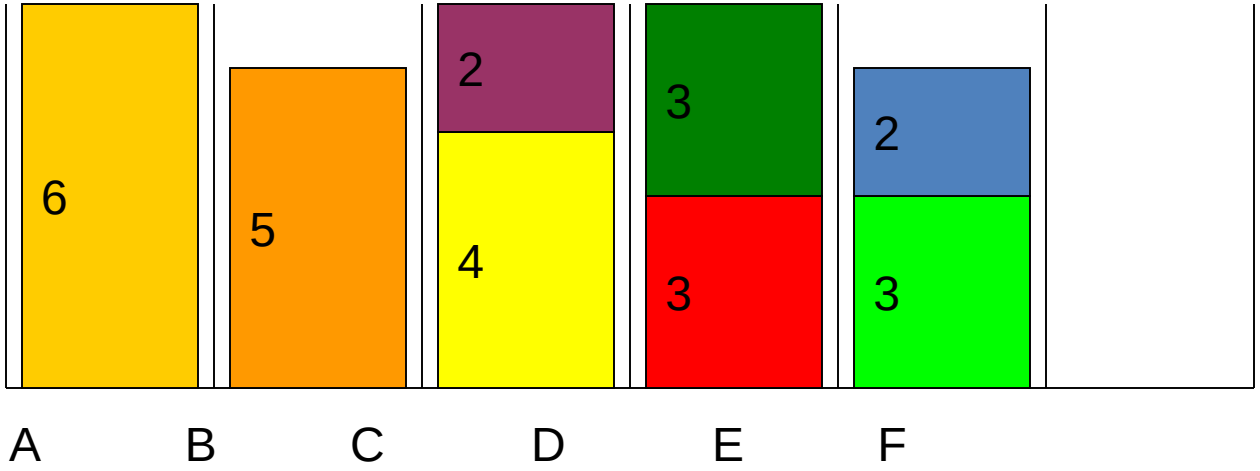
Bin Packing
First fit decreasing algorithm



Now we use the first fit algorithm

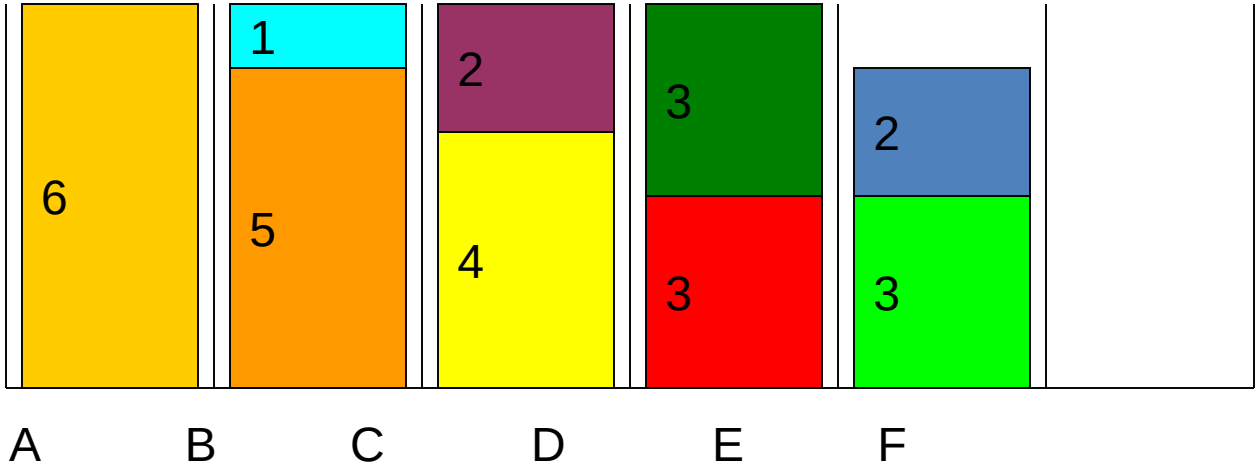


Bin Packing
First fit decreasing algorithm



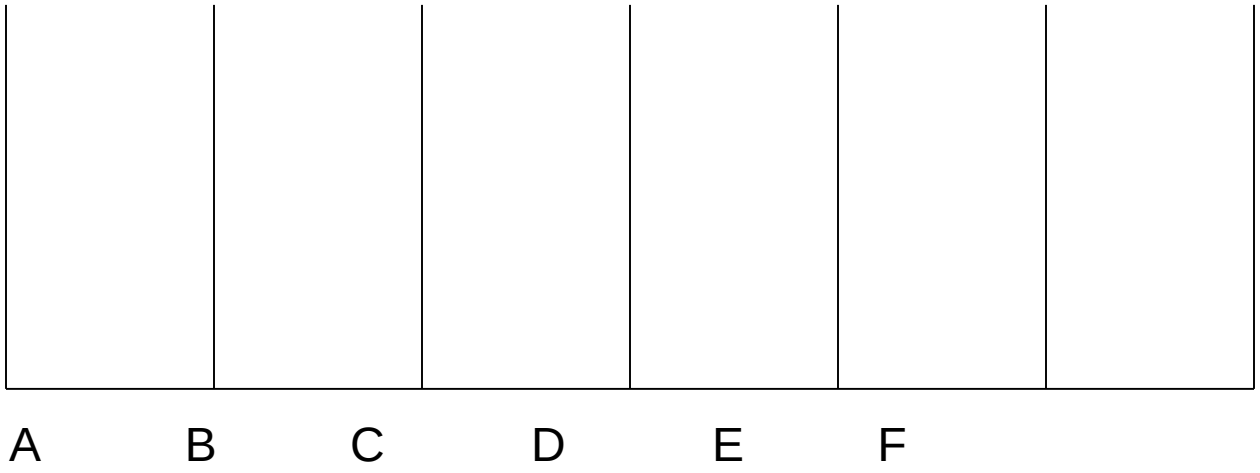
Now we use the first fit algorithm

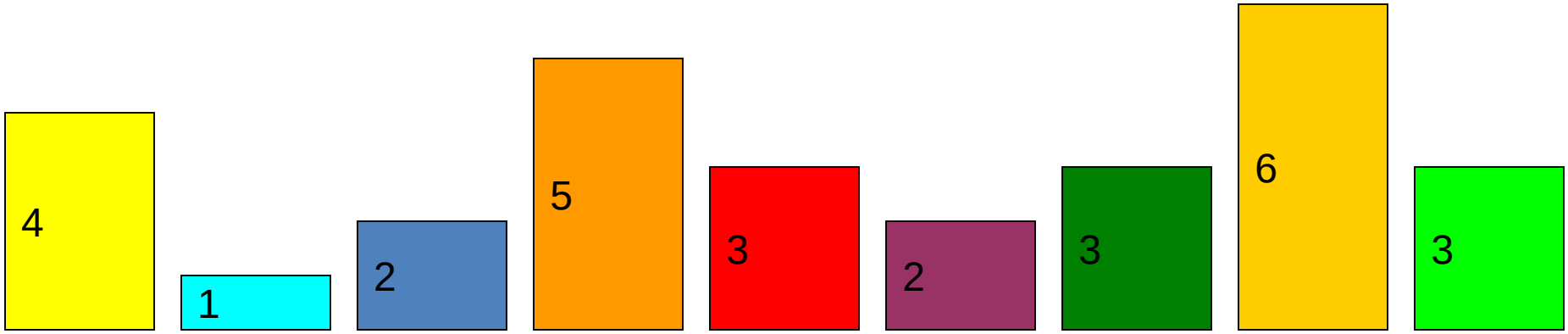
Bin Packing
First fit decreasing algorithm



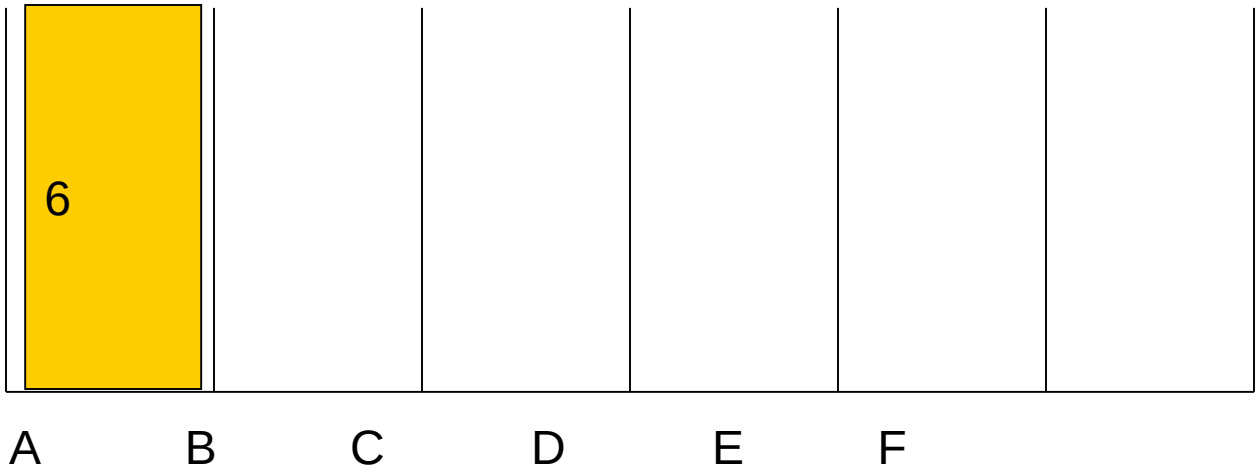
We have packed them into 5 bins.

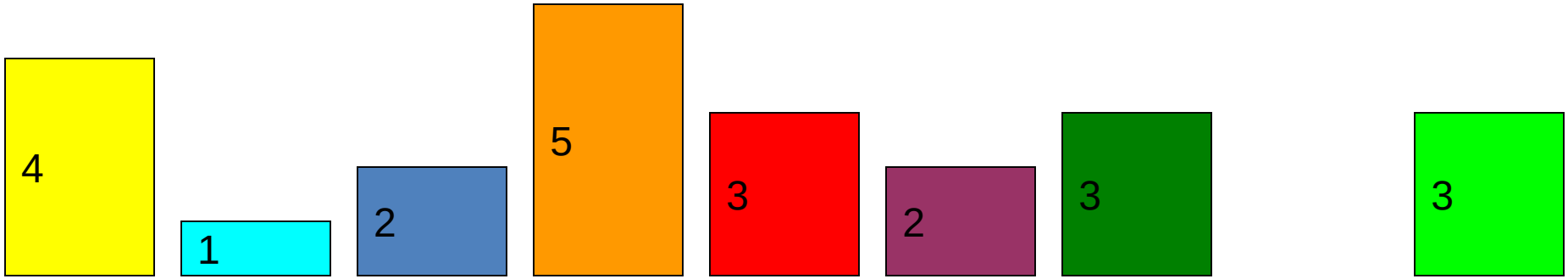
Bin Packing
Best fit algorithm



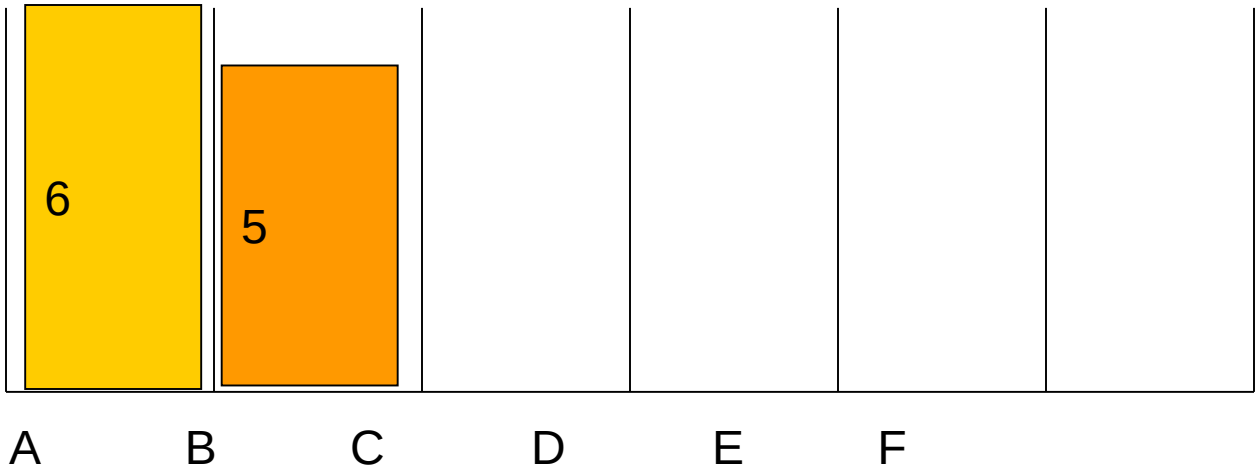


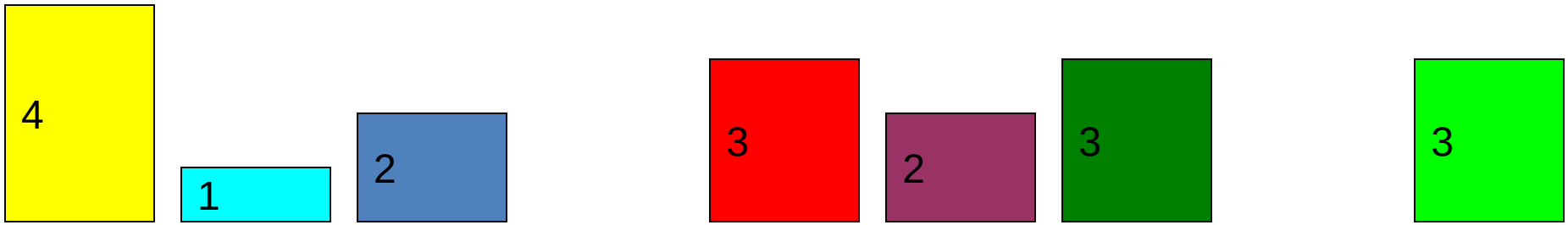
Bin Packing
Best fit algorithm



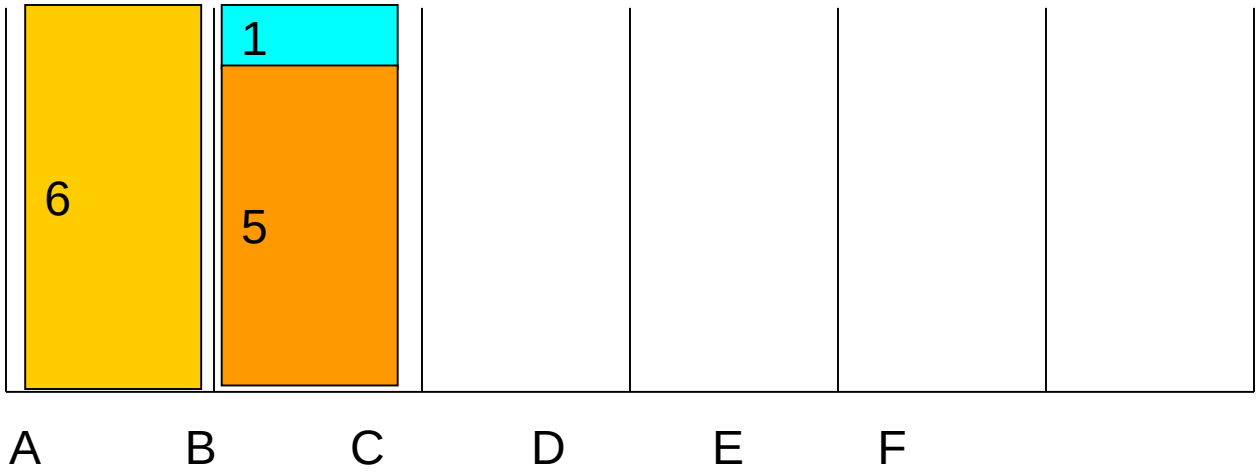


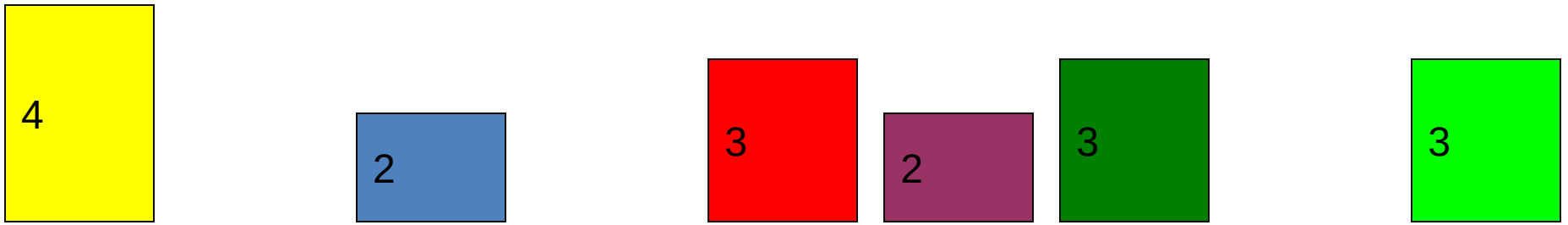
Bin Packing
Best fit algorithm



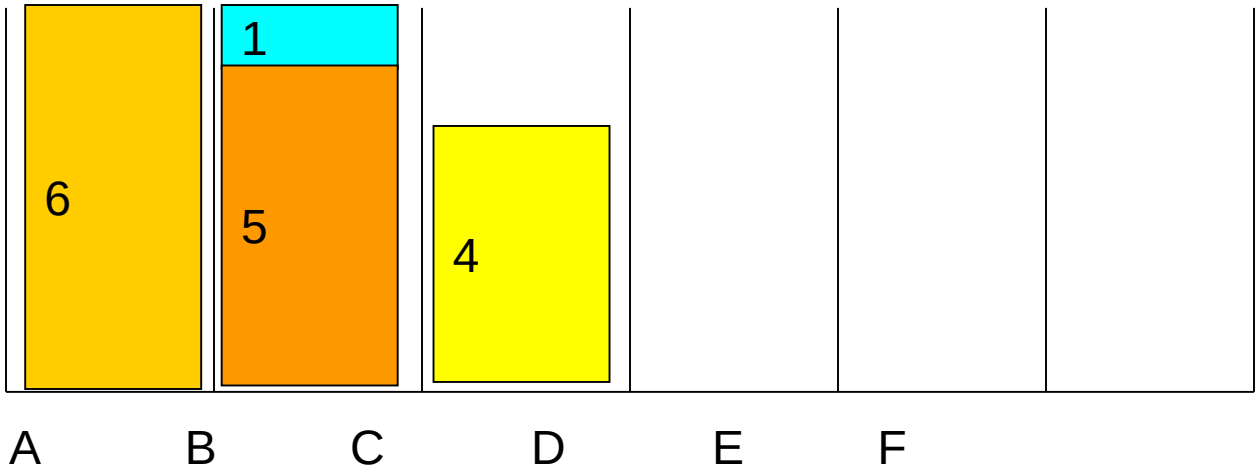


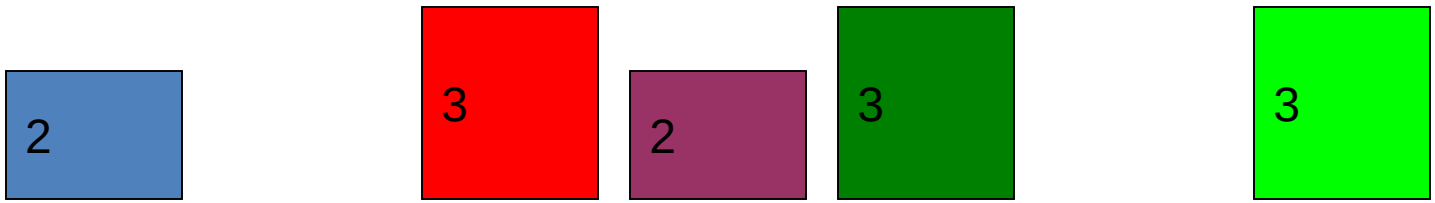
Bin Packing
Best fit algorithm



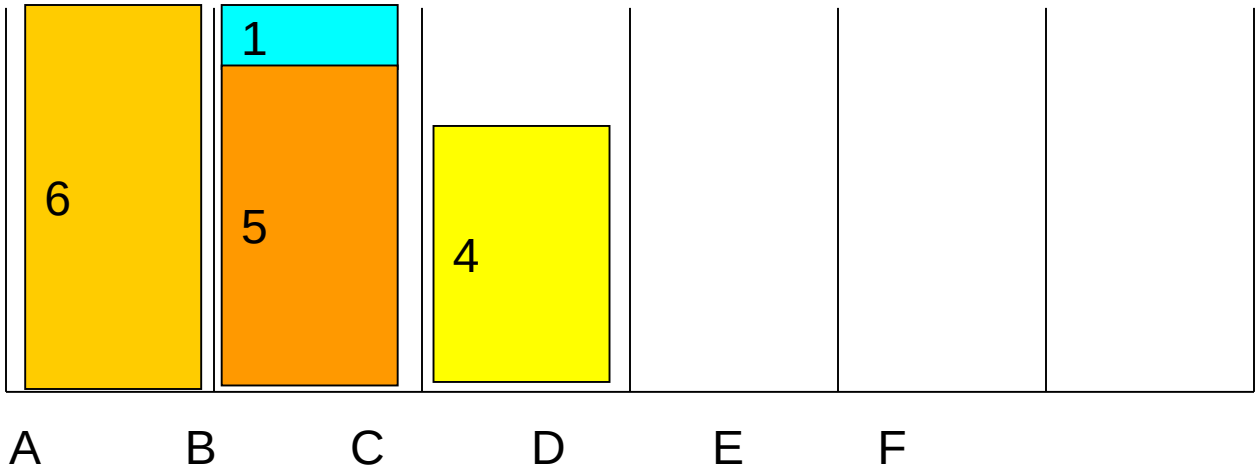


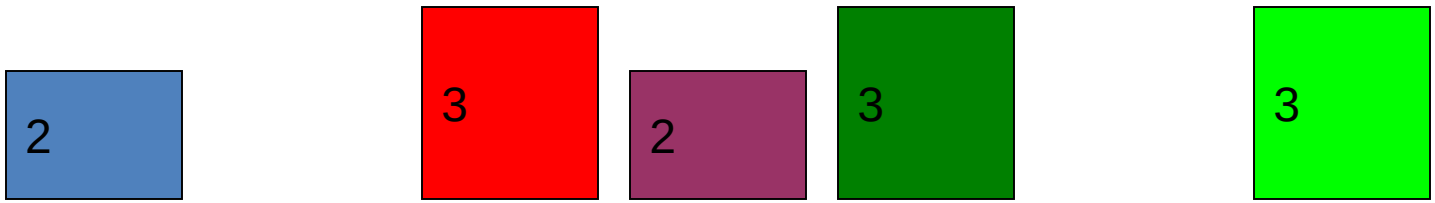
Bin Packing
Best fit algorithm



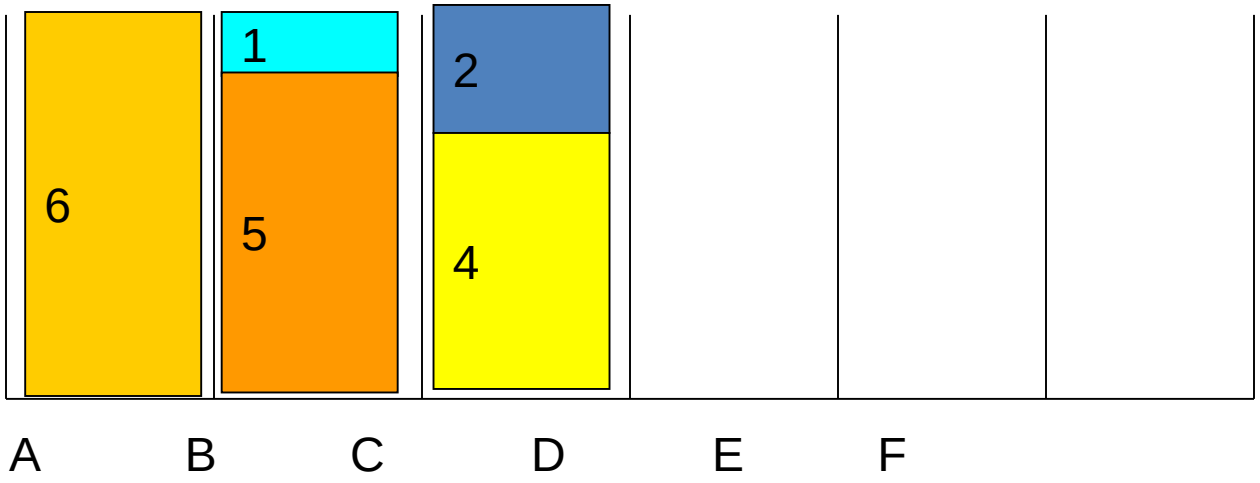


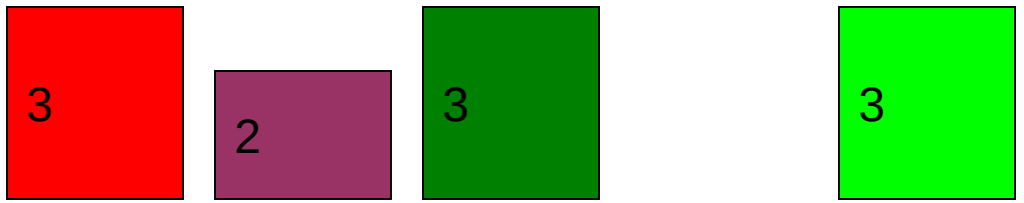
Bin Packing
Best fit algorithm



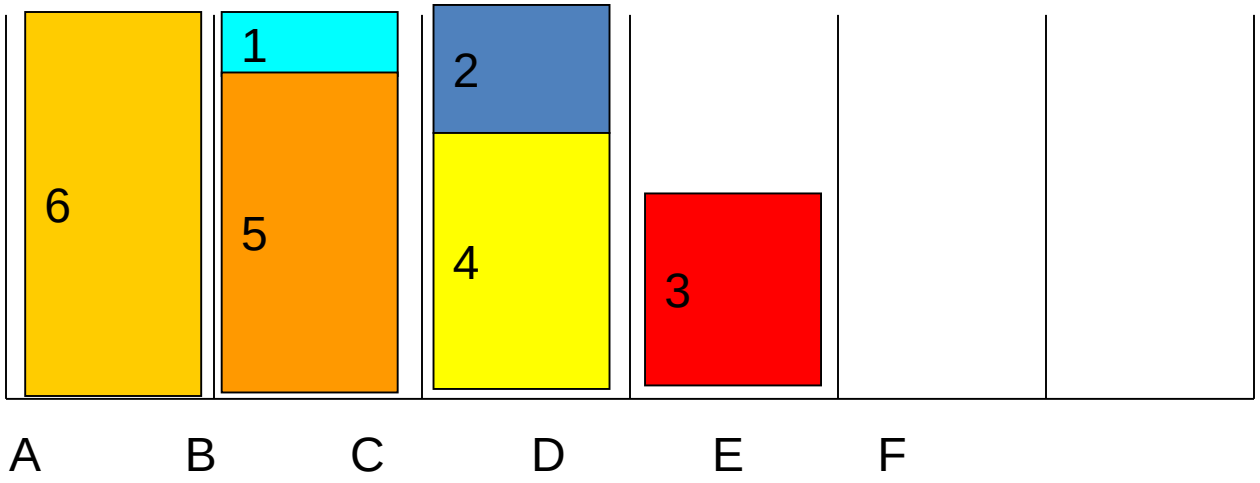


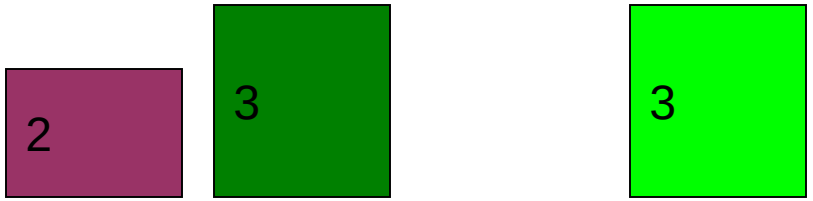
Bin Packing
Best fit algorithm



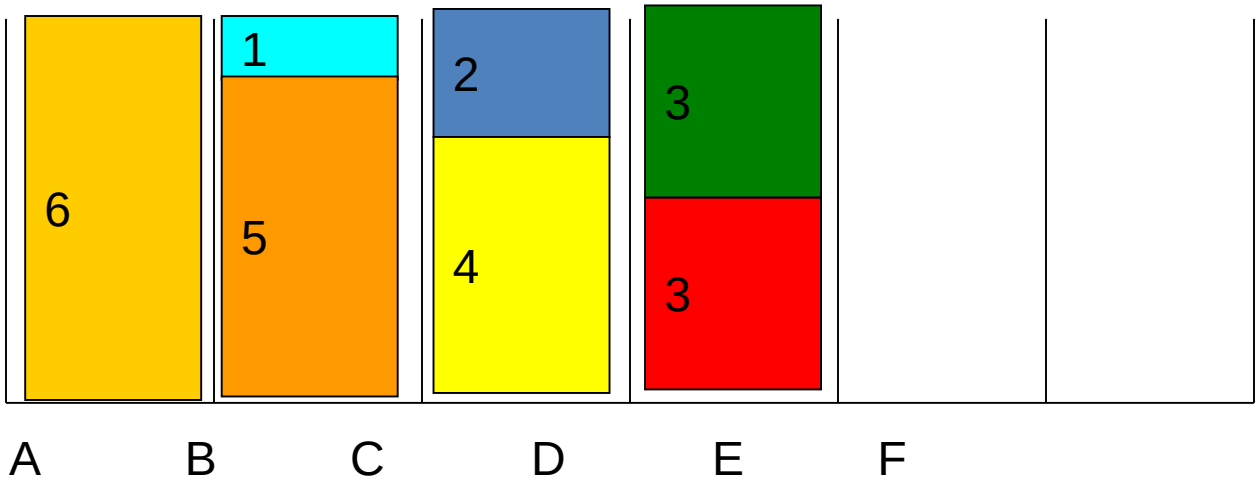


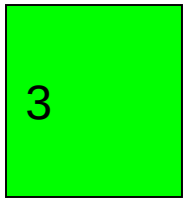
Bin Packing
Best fit algorithm



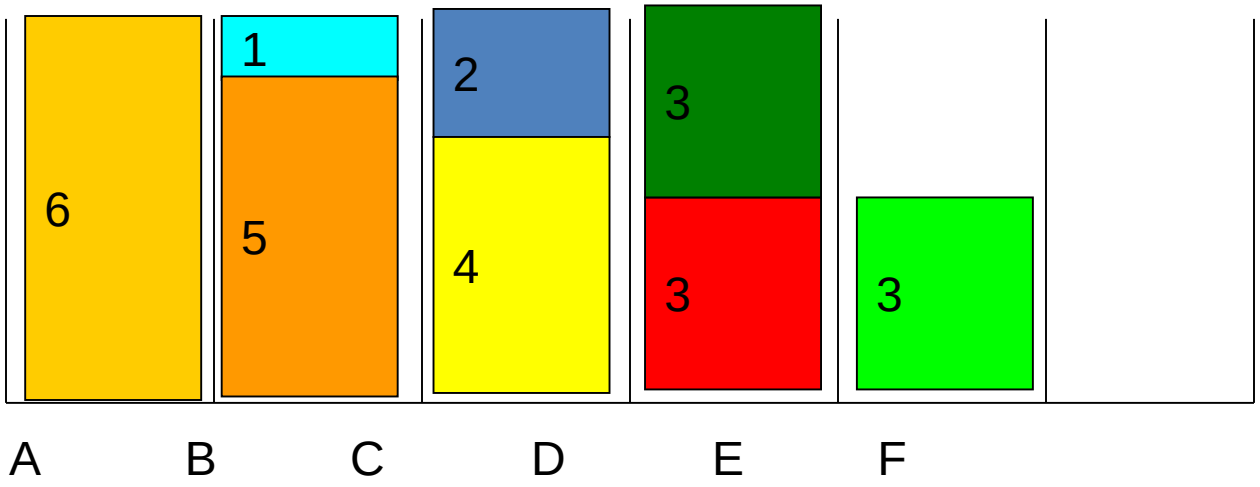


Bin Packing
Best fit algorithm

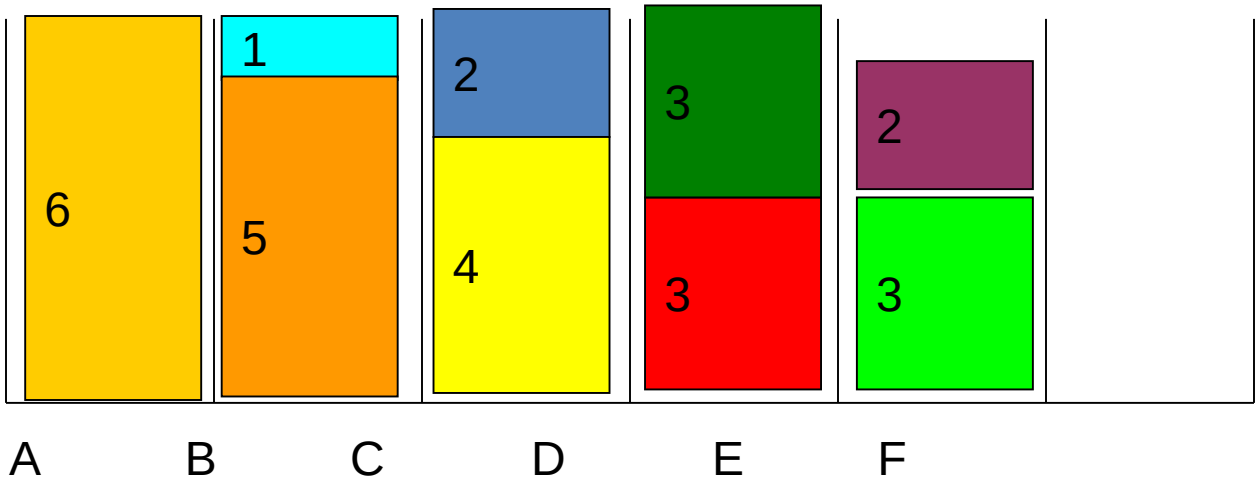




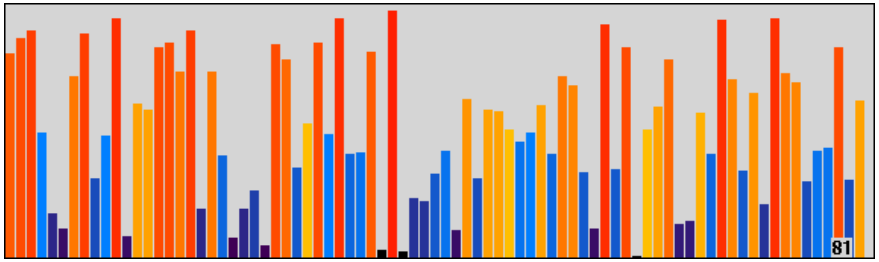
Bin Packing
Best fit algorithm



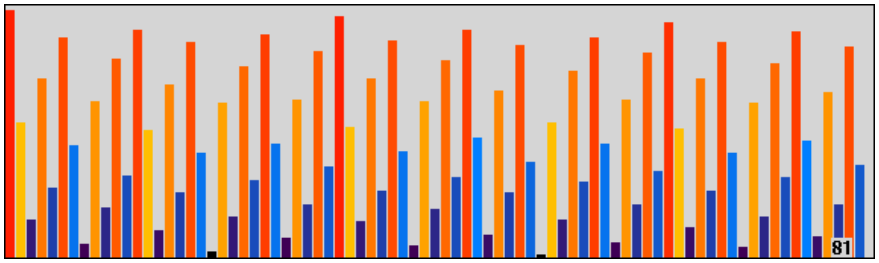
Bin Packing
Best fit algorithm



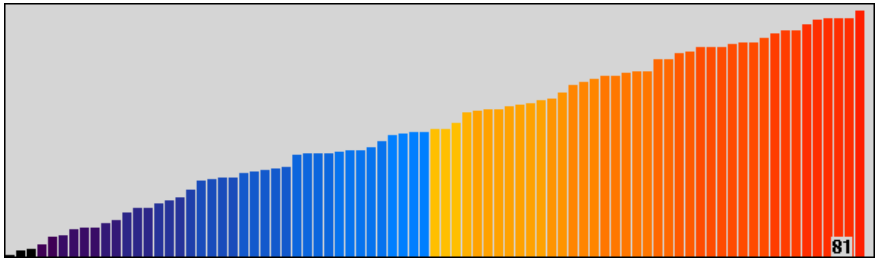
Синтез алгоритмов упаковки - Сортировка



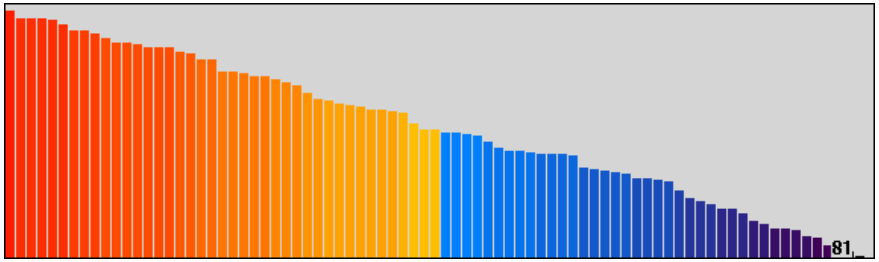
The **Random** entry reorders the candidate objects randomly.



The **Regular** entry maximizes the distances between similarly-sized objects, producing the pattern seen here.

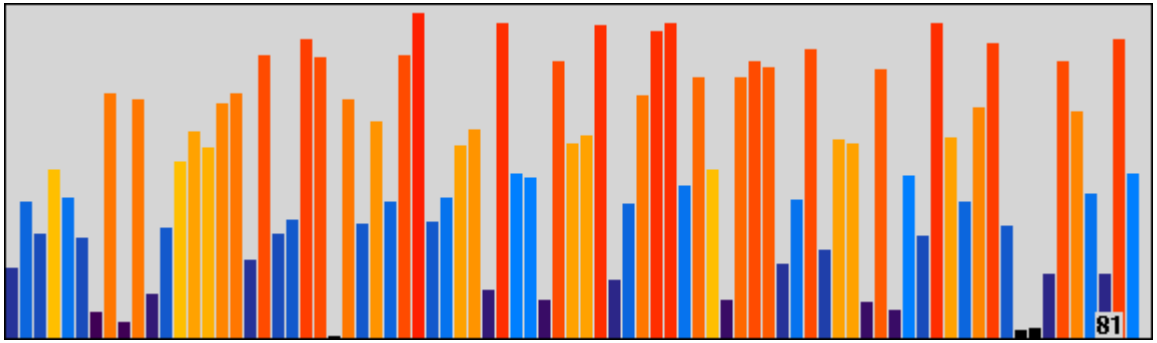


The **Ascending** entry orders the objects from smallest to largest. This display also illustrates the color scheme.

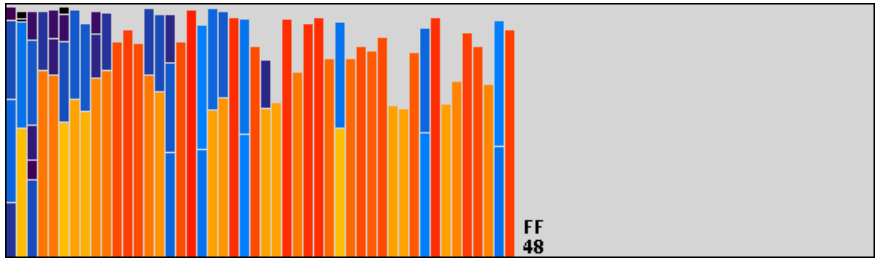


The **Descending** entry orders the objects from largest to smallest. Packing is usually improved by starting with the largest objects first.

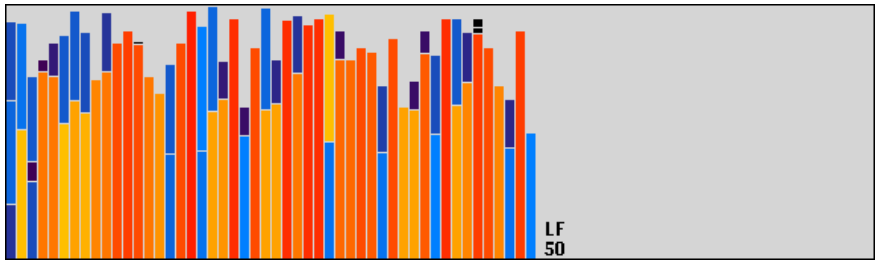
Синтез алгоритмов упаковки - Упаковка



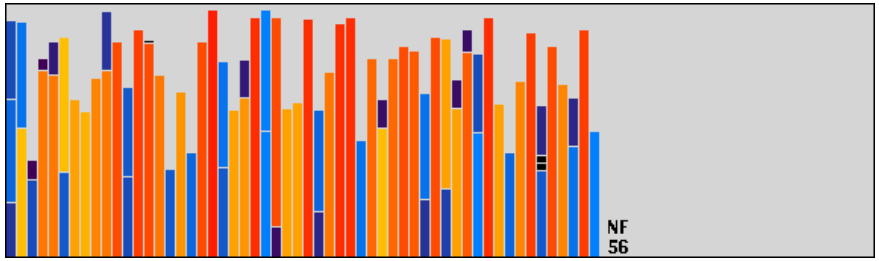
Синтез алгоритмов упаковки - Упаковка



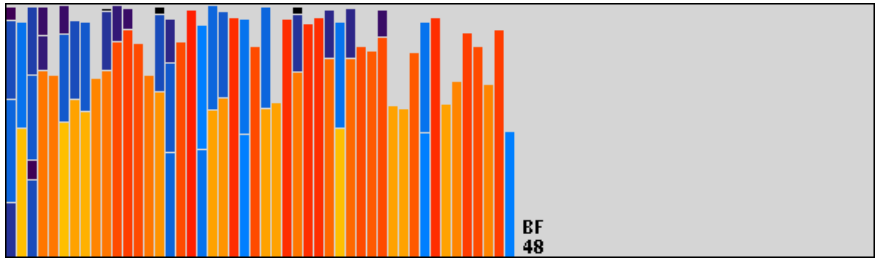
The **First Fit** algorithm places a new object in the leftmost bin that still has room.



The **Last Fit** algorithm places a new object in the rightmost bin that still has room.

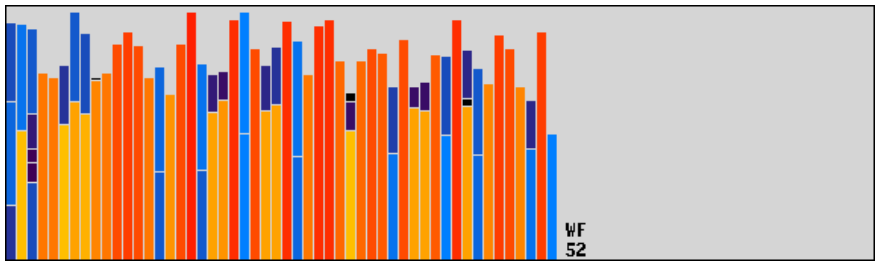


The **Next Fit** algorithm places a new object in the rightmost bin, starting a new bin if necessary.

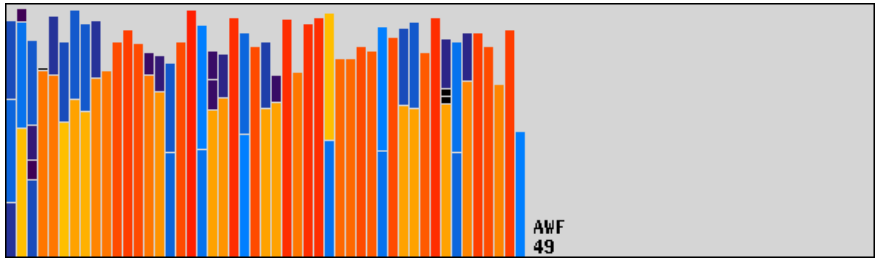


The **Best Fit** algorithm places a new object in the fullest bin that still has room.

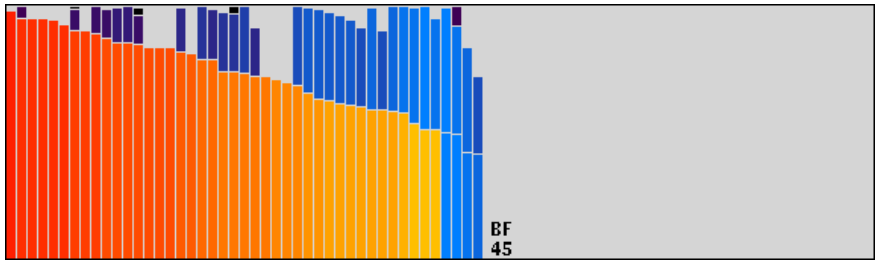
Синтез алгоритмов упаковки - Упаковка



The **Worst Fit** algorithm places a new object in the emptiest existing bin.



The **Almost Worst Fit** algorithm places a new object in the second-emptiest bin. (This algorithm is provably better than Worst Fit and so it is of some theoretical interest.)



Sorted Packing. In real applications, the sizes to be packed may all be known in advance. In that case, better results are achieved by packing the largest objects first. This illustration shows the results of sorting the input in descending order before applying the Best Fit algorithm.