National Taiwan University
Graduate Institute of Electronics Engineering
Physical Design for Nanometer ICs, Spring 2020

Handout #3
March 13, 2020
Yao-Wen Chang & Yu-Sheng Lu

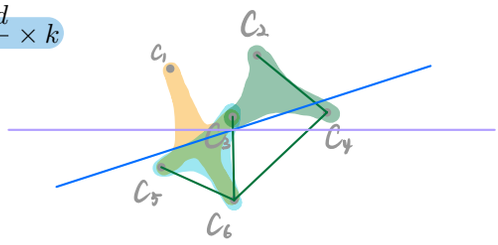
Programming Assignment #1: 2-Way F-M Circuit Partitioning; due 5pm on-line, April 4, 2020 (Saturday)

Modified from Problem #3 of the 2001 IC/CAD Contest (Source: Faraday Technology Corp.)

1. Problem Description

Let $C = \{c_1, c_2, c_3, ..., c_k\}$ be a set of k cells and $N = \{n_1, n_2, n_3, ..., n_m\}$ be a set of m nets. Each net n_i connects a subset of the cells in C. The 2-way partitioning problem is to partition the set C of k cells into two *disjoint, balanced* groups, G_1 and G_2, such that the overall cut size is minimized; in other words, no cell replication is allowed. The cut size s is given by the number of nets between G_1 and G_2. For a given *balance degree* d, $0 < d < 1$ (with $d = 0$ being the most balanced and $d = 1$ the least), the objective of this assignment is to minimize s under the constraint

$$\frac{1-d}{2} \times k \leq \#(G_1), \#(G_2) \leq \frac{1+d}{2} \times k$$



by the Fiduccia-Mattheyses heuristic introduced in class.

2. Input

The input format and a sample input are given as follows:

Input Format	Sample Input
<Balance Degree> NET <Net Name> [<Cell Name>]+;	0.5 NET n1 c2 c3 c4 ; NET n2 c3 c6 ; NET n3 c3 c5 c6 ; NET n4 c1 c3 c5 c6 ; NET n5 c2 c4 ; NET n6 c4 c6 ; NET n7 c5 c6 ;

The input file starts with the balance degree d, followed by the description of m nets. The description of each net contains the keyword NET, followed by the net name and a list of the connected cells, and finally the symbol ';'. See the sample input for the format of a circuit with seven nets and six cells.

3. Output

In the program output, you are asked to give the cut size, the sizes of $G1$ and $G2$, and the contents of $G1$ and $G2$. The following table gives the output format and an output to the sample input. (Note that the solution may not be the optimal one.)

Output Format	Sample Output
Cutsize = <Number> G1 <Size> [<Cell Name>]+; G2 <Size> [<Cell Name>]+;	Cutsize = 5 G1 3 c1 c2 c3 ; G2 3 c4 c5 c6 ;

4. Provided Program Structure

We provide some sample data and program structures for your reference. Below list several related files which are available at the submission web page (the file named "sample codes.tgz"):

Makefile
/bin
/src
/main.cpp: main program
/cell.h: cell and node information
/net.h: net information
/partitioner.h: header file for the partitioner
/partitioner.cpp: source code for the partitioner

To generate an executable binary, simply type "make" in the current directory. To regenerate a new executable binary, type "make clean" and then "make". Please note that you can add, delete, and/or revise any part of the sample codes. If so, please remember to revise your makefile accordingly. You are also recommended to construct your own program structure because it might be more difficult to understand others' programs than writing one from the scratch by yourself.

Command-line Parameter:
The executable binary must be named as **"fm"** and use the following command format:

```
./fm <input_file_name> <output_file_name>
```

For example, if you would like to run your binary for the input file `input_0.dat` and generate a solution named `output_0.dat`, the command is as follows:

```
./fm input_0.dat output_0.dat
```

Required Files:
You need to create a directory named **<student_id>_pa1/** (*e.g.* f08943000_pa1/) which must contain the following materials:

- A directory named **src/** containing your source codes: only `*.h`, `*.hpp`, `*.c`, `*.cpp` are allowed in src/, and no directories are allowed in src/;
- A directory named **bin/** containing your executable binary named **fm**;
- A makefile named **makefile or Makefile** that produces an executable binary from your source codes by simply typing "make": the binary should be generated under the directory <student_id>_pa1/;
- A text readme file named **readme.txt** describing how to compile and run your program.
- A report named **report.pdf** on the data structures used in your program and your findings in this programming assignment.

Then please use the following command to compress your directory into a *.tgz* file:

```
tar zcvf <filename>.tgz <your directory>
```

The submission file should be named as **<student_id>_pa1.tgz** (*e.g.* f08943000_pa1.tgz). For example, if your student ID is *f08943000*, then you should use the command below:

```
tar zcvf f08943000_pa1.tgz f08943000_pa1/
```

Please submit your *.tgz* file to the NTU COOL system before **5pm, April 4, 2020 (Saturday)**. To make sure that your submission satisfies all the requirements, we provide a script `checkSubmitPA1.sh`

to check your *.tgz* file by the command below:

```
bash checkSubmitPA1.sh <your submission>
```

For example,

```
bash checkSubmitPA1.sh f08943000_pa1.tgz
```

Language/Platform:

(a) Language: C or C++.

(b) Platform: Linux. (For fair evaluation of the submitted program, your developed programs will be tested on EDA Union servers. Submitted programs that fail to be executed on these servers by the TA will incur significant penalty.)

5. **Grading Policy**

   This programming assignment will be graded based on (1) **correctness**, (2) **solution quality**, (3) **running time**, (4) **readme.txt**, and (5) **report.pdf**. Please check these items before your submission. The runtime limit for each case is 60 minutes. For fair evaluation, please apply the **-O3** optimization for the compilation.

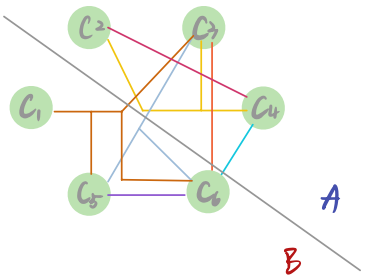
6. **Online Resources**

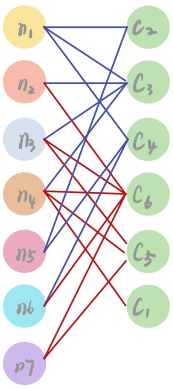
   Sample input files (input.dat), readme.txt, report.pdf, and a checker to verify your program results can be found at the NTU COOL course link below: https://cool.ntu.edu.tw/courses/1331/assignments/6355

   For any questions, please email Yu-Sheng Lu at yslu@eda.ee.ntu.edu.tw. Thank you so much for your cooperation. Have fun with this programming assignment!

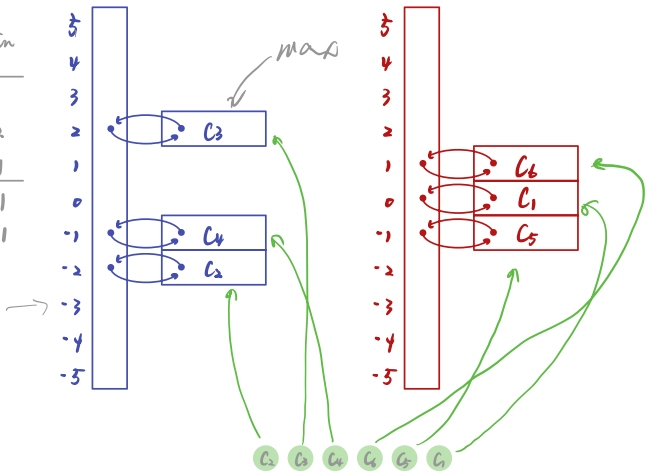
0.5

```
NET   n1   c2   c3   c4   ;
NET   n2   c3   c6   ;
NET   n3   c3   c5   c6   ;
NET   n4   c1   c3   c5   c6   ;
NET   n5   c2   c4   ;
NET   n6   c4   c6   ;
NET   n7   c5   c6   ;
```

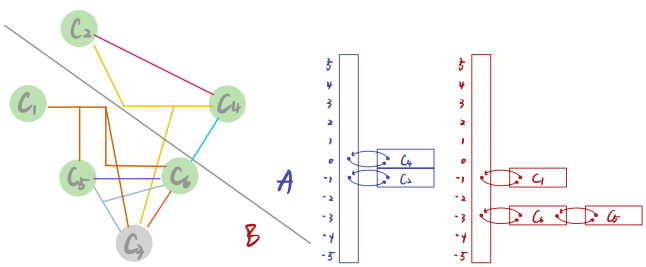




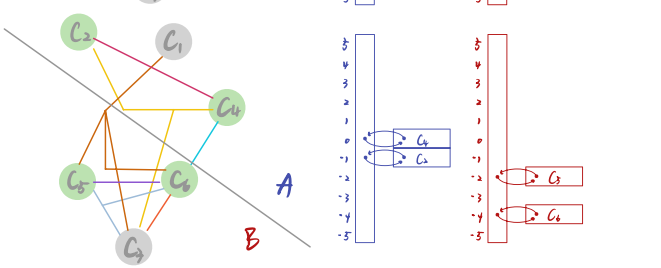
net \ cell	n1 F	n1 T	n2 F	n2 T	n3 F	n3 T	n4 F	n4 T	n5 F	n5 T	n6 F	n6 T	n7 F	n7 T	Gain
c2	0	-1							0	-1					-2
c3	0	-1	+1	0	+1	0	+1	0							+2
c4	0	-1							0	-1	+1	0			-1
c6			+1	0	0	0	0	0			+1	0	0	-1	+1
c5					0	0	0	0					0	-1	-1
c1							0	0							0



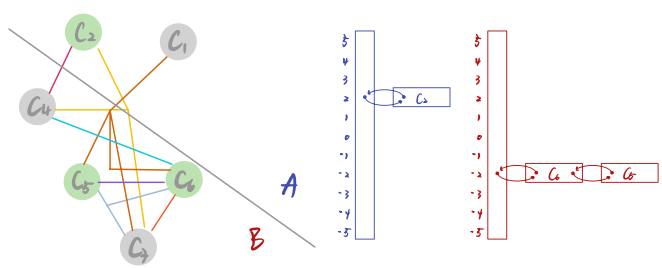
net \ cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain old	Gain new
c2	+1							0							-2	-1
c3															+2	+2
c4	+1							0							-1	0
c6		-1	-1	-1					-1	0	0				+1	-3
c5			-1	-1						0	0				-1	-3
c1				-1							0				0	-1



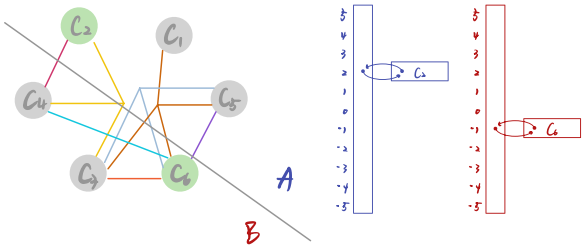
net \ cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain old	Gain new
c2															-1	-1
c3															+2	+2
c4															0	0
c6			+1												-3	-2
c5			+1												-3	-2
c1															-1	-1



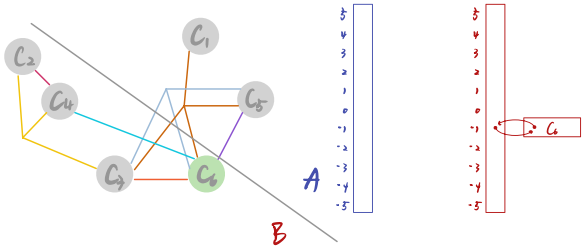
net \ cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain old	Gain new
c2															-1	+2
c3															+2	+2
c4															0	0
c6															-2	-4
c5															-2	-2
c1															-1	-1



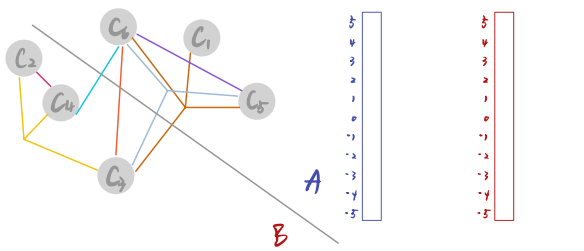
cell \ net	Gains due to $T(n)$							Gains due to $F(n)$							Gain	
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	old	new
c_3															+2	+2
c_2															+2	+2
c_4															0	0
c_6															-2	-1
c_5															-2	-2
c_1															-1	-1

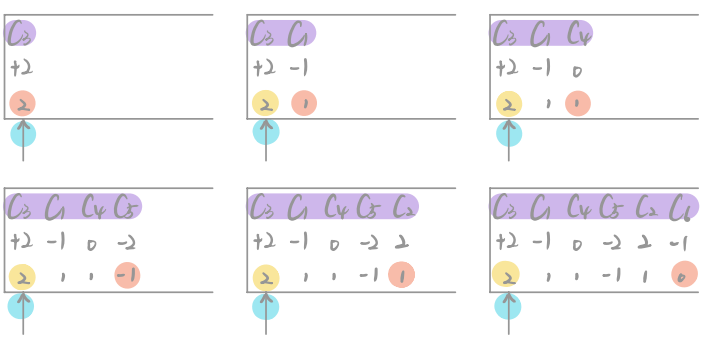


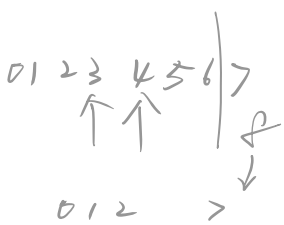
cell \ net	Gains due to $T(n)$							Gains due to $F(n)$							Gain	
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	old	new
c_3															+2	+2
c_2															+2	+2
c_4															0	0
c_6															-1	-1
c_5															-2	-2
c_1															-1	-1



cell \ net	Gains due to $T(n)$							Gains due to $F(n)$							Gain	
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	old	new
c_3															+2	+2
c_2															+2	+2
c_4															0	0
c_6															-1	-1
c_5															-2	-2
c_1															-1	-1







partitioner

partSize[0] partSize[1]

Cell

part

net

partCount[0] partCount[1]

不一樣的發

cutSize 有增有減

cutSize +2

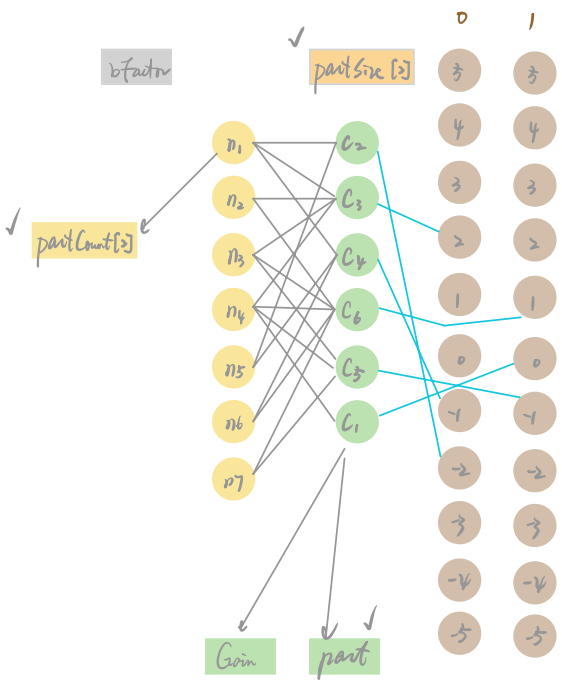
一樣的發

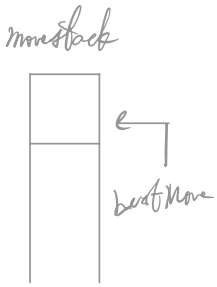
unordered_map

map< string , tuple< vector< string >, vector< unsigned > > > net

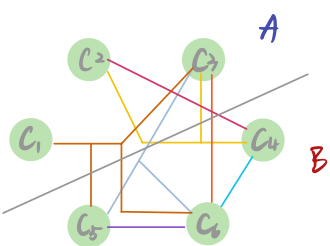
map< string , tuple< vector< string >, list< unsigned >::iterator , int , bool > > cell

vector< map< int , list< string > > > gainBucket



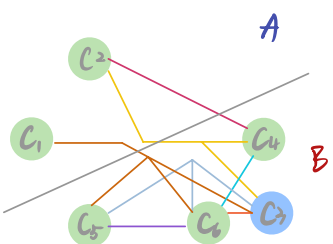


net \ cell	n1 F	n1 T	n2 F	n2 T	n3 F	n3 T	n4 F	n4 T	n5 F	n5 T	n6 F	n6 T	n7 F	n7 T	Gain
c1							0	0							0
c2	0	0							+1	0					+1
c3	0	0	+1	0	+1	0	0	0							+2
c4	+1	0							+1	0	0	-1			+1
c5					0	0	0	0					0	-1	-1
c6			+1	0	0	0	0	0	0	-1	0	-1			-1



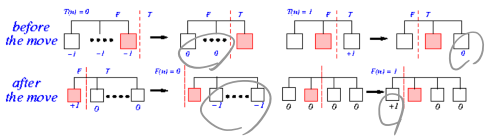

```
0.5
NET n1 c2 c3 c4 ;
NET n2 c3 c6 ;
NET n3 c3 c5 c6 ;
NET n4 c1 c3 c5 c6 ;
NET n5 c2 c4 ;
NET n6 c4 c6 ;
NET n7 c5 c6 ;
```

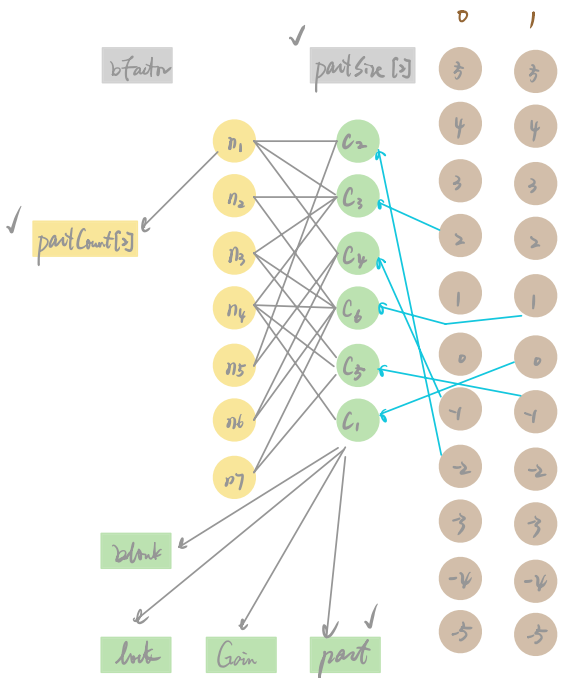
net \ cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain
c1											+1				+1
c2								+1							+2
c3															+2
c4	-1														0
c5											-1				-2
c6		-1									-1	-1			-4



update()
delete bucket
part
partSize
partCount
update gain
delete from iterator
insert
iterator link

net \ cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain
c1															
c2															
c3															+2
c4															
c5															






```
vector<string> netName;
vector<vector<unsigned>> netEdge;
vector<vector<unsigned>> partCount;

vector<string> CellName;
vector<vector<unsigned>> cellEdge;
vector<int> gain;
vector<bool> part;
vector<bool> lock;
vector<list<unsigned>::iterator> blink
```

~~vector<map<int, list<unsigned>>>~~ gainBucket[2]
BucketList

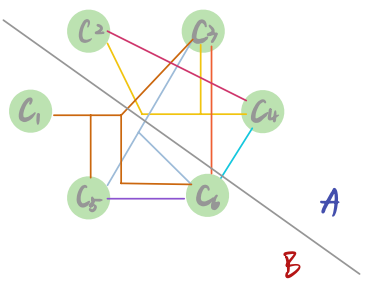
```
double bFactor
vector<unsigned> partSize
unsigned pmax;
```

0.5

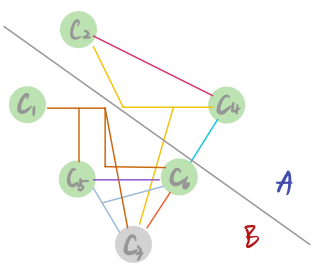
```
NET n1 c2 c3 c4 ;
NET n2 c3 c6 ;
NET n3 c3 c5 c6 ;
NET n4 c1 c3 c5 c6 ;
NET n5 c2 c4 ;
NET n6 c4 c6 ;
NET n7 c5 c6 ;
```

bucket 空
all lock ✓
part ✓
gain ✗
blink ✗
partCount ✓
partSize ✓

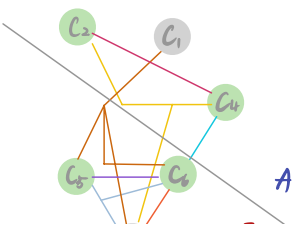
net / cell	n1 F	n1 T	n2 F	n2 T	n3 F	n3 T	n4 F	n4 T	n5 F	n5 T	n6 F	n6 T	n7 F	n7 T	Gain
c2	0	-1							0	-1					-2
c3	0	-1	+1	0	+1	0	+1	0							+2
c4	0	-1							0	-1	+1	0			-1
c6			+1	0	0	0	0	0			+1	0	0	-1	+1
c5					0	0	0	0					0	-1	-1
c1							0	0							0



net / cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain
c2															-1
c3															+2
c4															0
c6															-3
c5															-3
c1															-1

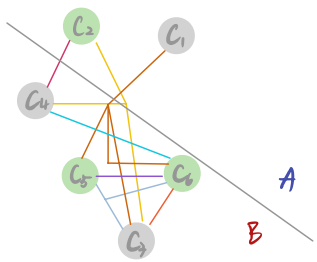


net / cell	Gains due to T(n) n1	n2	n3	n4	n5	n6	n7	Gains due to F(n) n1	n2	n3	n4	n5	n6	n7	Gain
c2															-1
c3															+2
c4															0
c6															-2
c5															-2

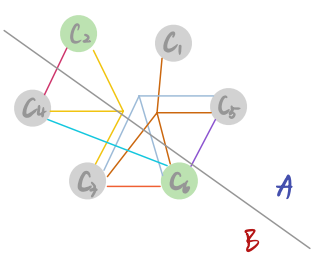


c_1			−1

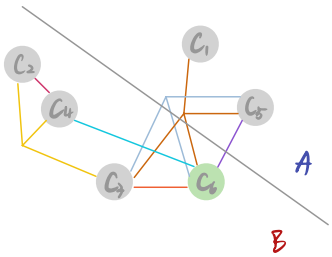
net / cell	Gains due to $T(n)$							Gains due to $F(n)$							Gain
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	
c_2															+2
c_3															+2
c_4															0
c_6															−4
c_5															−2
c_1															−1



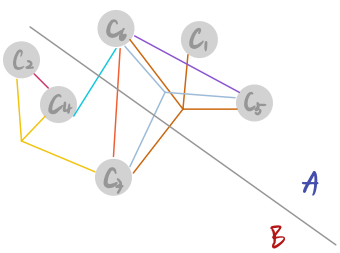
net / cell	Gains due to $T(n)$							Gains due to $F(n)$							Gain
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	
c_2															+2
c_3															+2
c_4															0
c_6															−1
c_5															−2
c_1															−1

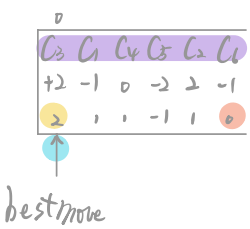


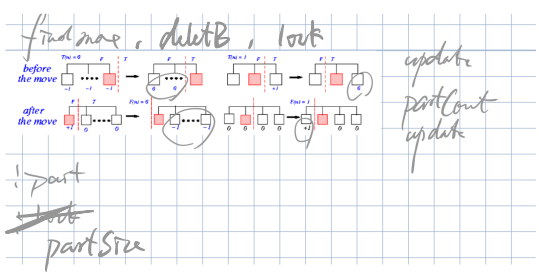
net / cell	Gains due to $T(n)$							Gains due to $F(n)$							Gain
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	
c_2															+2
c_3															+2
c_4															0
c_6															−1
c_5															−2
c_1															−1



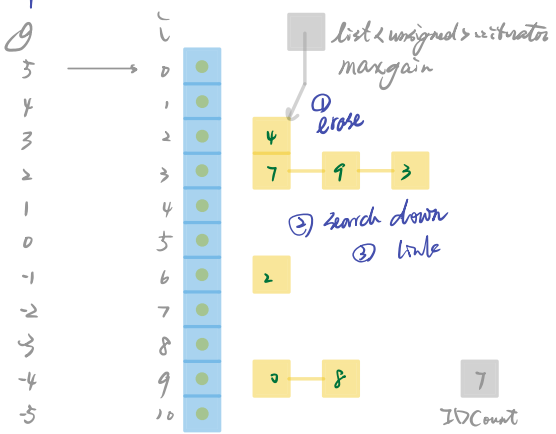
net / cell	Gains due to $T(n)$							Gains due to $F(n)$							Gain
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	
c_2															+2
c_3															+2
c_4															0
c_6															−1
c_5															−2
c_1															−1

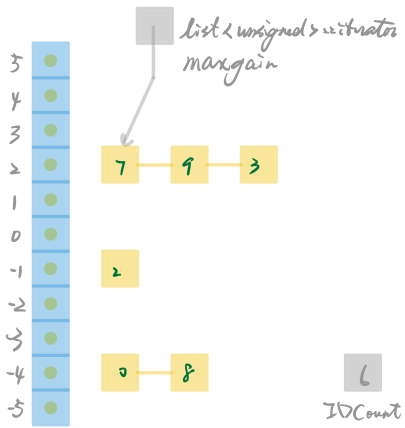






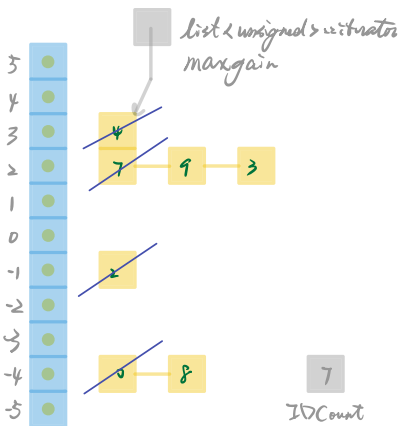
operator[i] unsigned popmax(void);

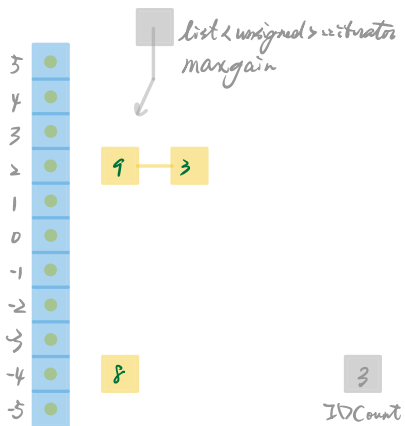




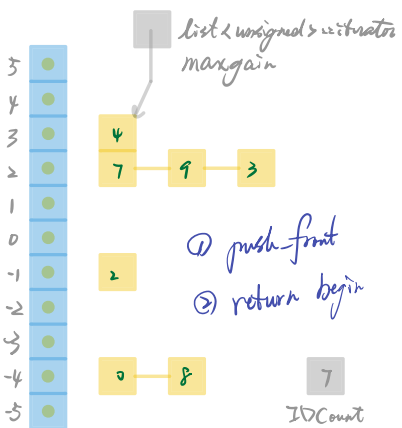
5 -g=i

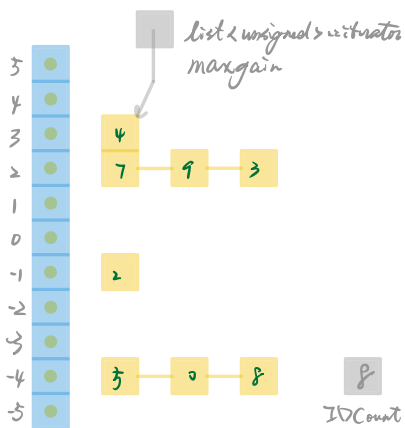
void remove(gain, iterator)





iterator add(gain)





cell \ net	Gains due to $T(n_i)$							Gains due to $F(n_i)$							Gain
	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_1	n_2	n_3	n_4	n_5	n_6	n_7	
c_2															
c_3															
c_4															
c_6															
c_5															
c_1