Physical Design HW2

Problem 1 [Collaborator: None]

The simplify pseudo code of the SA-based partition algorithm is shown in the following:

```
Initialize parameters:
    T_end, max_reject_ratio, reduce_factor, steps_per_iteration
A = Set of first falf of the cells
B = Set of the remain cells not included in set A
Partition = {A, B} // initial solution (must be balanced)
Best_Solution = Partition
cur_cost = Cut size of the current partition
repeat
    reject = 0
    move_num = 0
    repeat
        Current_Solution = Random choose k Cells from any side
        if(the Current_Solution is unbalanced) // if unbalanced, skip this move
            continue;
        next_cost = Cost of current partition;
        if(next_cost > cur_cost) // downhill move
            cur_cost = next_cost;
            Best_Solution = Current_Solution
        else if(Rand()<e^(-(cur_cost-next_cost)/T)) // take uphill move
            cur_cost = next_cost;
        else // reject move, backtrack the solution
            ++reject;
            Move the k Cell back;
        ++move_num;
    until (move_num > steps_per_iteration)
    T = reduce_factor*T;  // reduce temperature
until (reject/move_num > max_reject_ratio) or (T < T_end) or OutOfTime;
```

We can see that 4 basic ingredients are in this algorithm:
(1) Solution space: All balanced partitions.
(2) Neighborhood structure: The neighborhood structure is the balanced partition set by randomly moving k cells from one side to the other.
(3) Cost function: The cost function is the cut size of the bi-partition.
(4) Annealing schedule: Non-zero probability for uphill move, and the temperature is reduce by a factor for each iteration.

Problem 2 [Collaborator: None]

Given the expresison: $E = 12V34HVH5$
a.

index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
literal	1	2	V	3	4	H	V	H	5
# operators	0	0	1	1	1	2	3	4	4
# operands	1	2	2	3	4	4	4	4	5

We can see that in index 8, the property (# operands > # operators) is not satisfy. Therefore, the expression does not have the balloting property.

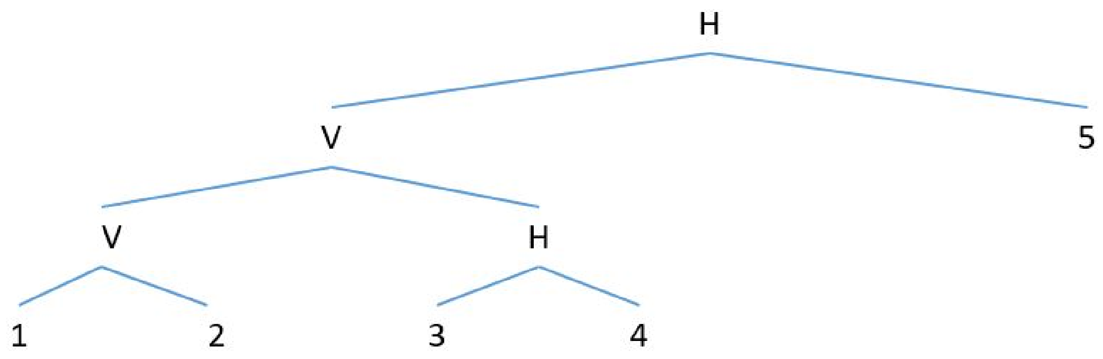
b.

The expression E is not a normalized polish expression, since it is not even a Polish expression, not satisfying the balloting property.

If we swap the operator in index 8 and the operands in index 9, then the new expression $E' = 12V34HV5H$ is a polish expression. Moreover, there are no consecutive operators of the same type, so E' is a normalized polish expression.

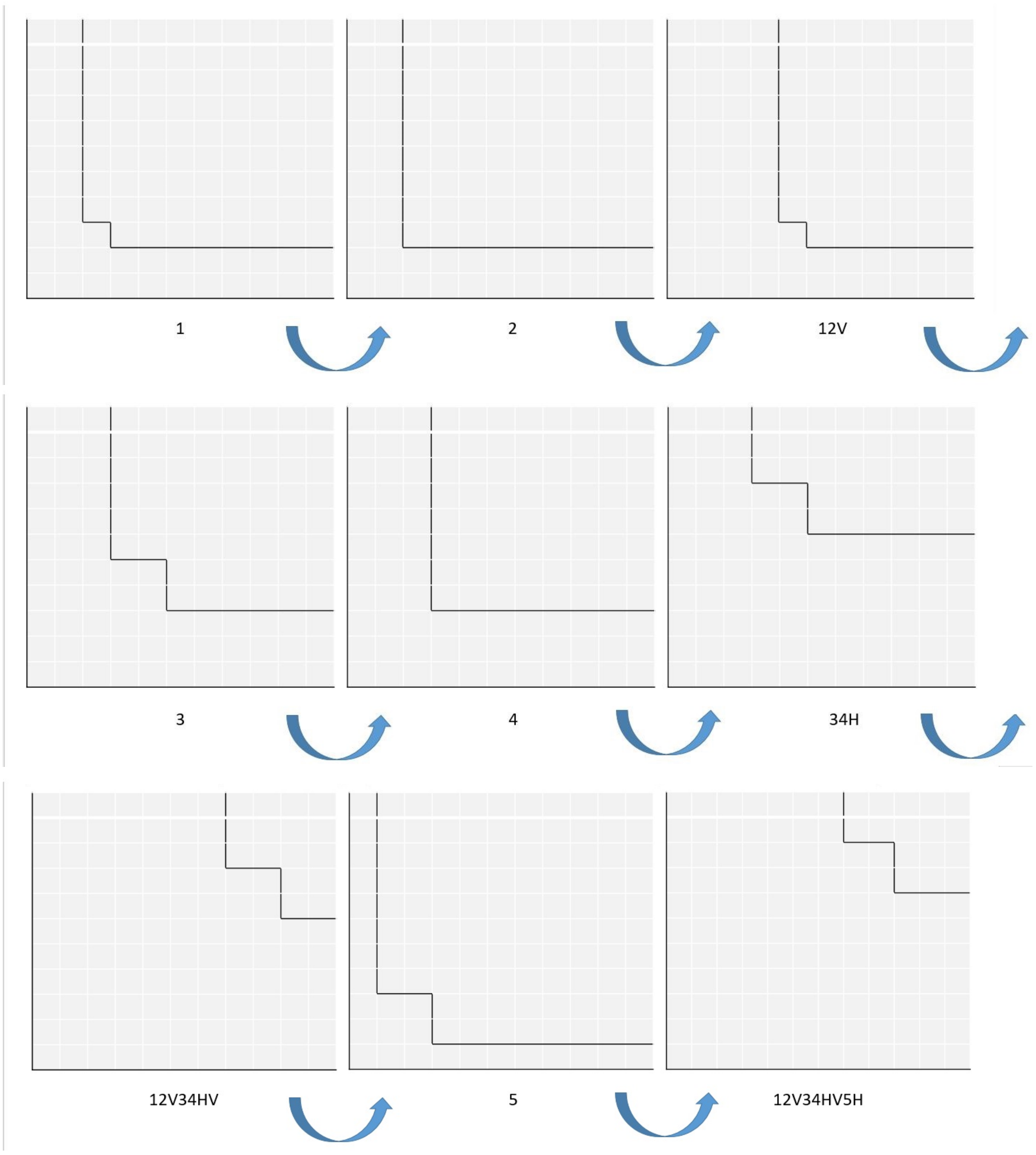
c.

The slicing tree of the normalized polish expression E' is shown in the figure below.



d.

We can calculate the smallest bounding rectangle by drawing the “Shape Curve” for the expression in each step.



From the final “Shape Curve”, we can see that the possible solution is 7×9 (63) and 9×7 (63). Both of them are the smallest packed area of this slicing tree.

Problem 3 [Collaborator: None]

For the sequence $A = \{(p_1, q_1), (p_2, q_2), \dots, (p_m, q_m)\}$ and a given point (x_k, y_k) in sequence B, we can find an index l such that $q_1 \geq q_2 \geq \dots \geq q_l \geq y_k \geq q_{l+1} \geq \dots \geq q_m$.

We can show that: $(p_{l+1} + x_k, max\{q_{l+1}, y_k\})$ dominates $(p_i + x_k, max\{q_i, y_k\})$ for $i = (l+1) \sim m$. Since $max\{q_i, y_k\} = y_k$ and $p_i + x_k \geq p_l + x_k$ for $i = (l+1) \sim m$.

From this property we can see that the elements chosen in C are either **chain** of elements in A or B. Where the definition of **chain** of a element y_i in a set is that the longeset consecutive sequence in the other set such that the property $y_{i-1} \geq q_{l_i-1} \geq \dots \geq q_{l_i-1} \geq y_i$ holds.

For example, if we have $A = \{(x_1, y_1), \dots, (x_9, y_9)\}$ and $B = \{(p_1, q_1), \dots, (p_5, q_5)\}$, and the order after sorting $A \cup B$ by decresing y value is $(x_1, y_1), \dots, (x_5, y_5), (p_1, q_1), \dots, (p_4, q_4), (x_6, y_6), \dots, (x_9, y_9), (p_5, q_5)$, then the chain of (p_1, q_1) is $(x_1, y_1), \dots, (x_5, y_5)$,

the chain of (x_6, y_6) is $(p_1, q_1), \ldots, (p_4, q_4)$,
the chain of (p_5, q_5) is $(x_6, y_6), \ldots, (x_9, y_9)$,
and the other remain elements has no chain.

Since the number of chains of all elements is at most $m + n - 1$, the number of elements in C is linear function of $m + n$.

Problem 4 [Collaborator: None]

a.

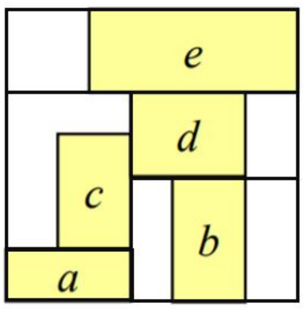


Fig. A

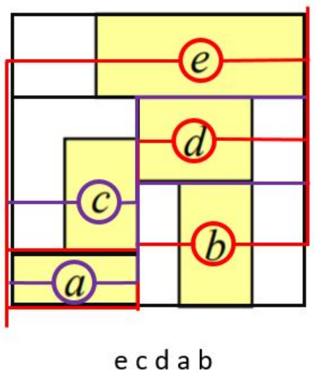


Fig. B

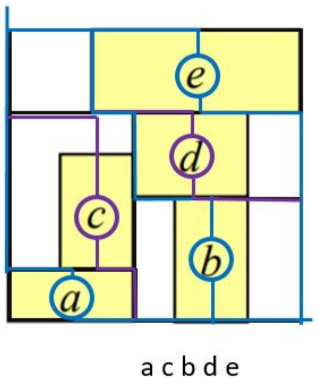


Fig. C

First, we partition the floorplan into rooms, each containing at most one block (Fig. A)
Second, we form the positive locus $\Gamma_+ = ecdab$ (Fig. B)
Third, we form the negative locus $\Gamma_- = acbde$ (Fig. C)
b.

First, for each pair of block, we can define either H-constrainst or V-constriant of it, shown in the table below.

	a	**b**	**c**	**d**	**e**
a	-	a_{left}	a_{bottom}	a_{bottom}	a_{bottom}
b	-	-	b_{right}	b_{bottom}	b_{bottom}
c	-	-	-	c_{left}	c_{bottom}
d	-	-	-	-	d_{bottom}
e	-	-	-	-	-

Then, we can form the edge based on these constrainst and get Fig.E and Fig.F.
Next, we remove the transitive edge and add the supersource, connecting to the node with no previous node, and supersink, connecting to the node with no next node. The resulting Horizontal Constrainst graph G_H is shown in Fig.G and the Vertical Constrainst graph G_V is shown in Fig.H.

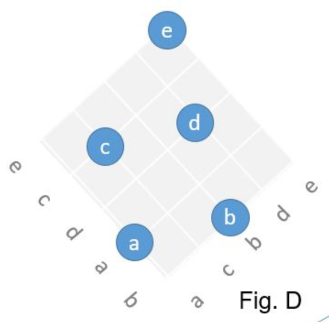


Fig. D

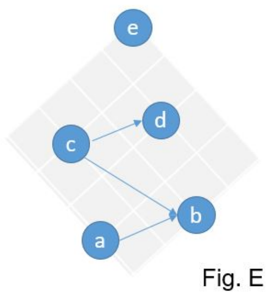


Fig. E

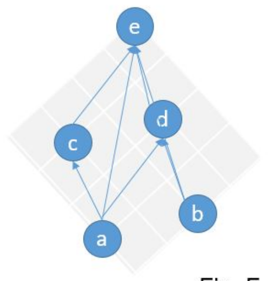


Fig. F

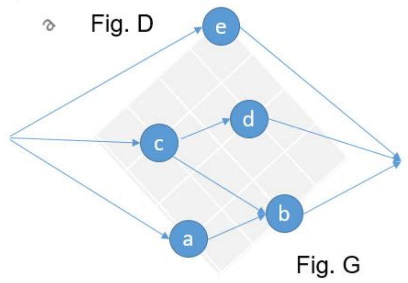


Fig. G

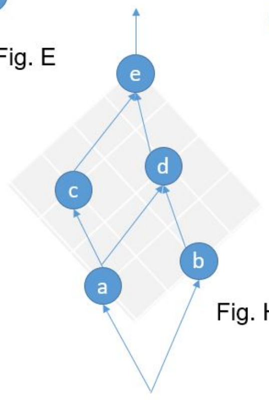


Fig. H

c.

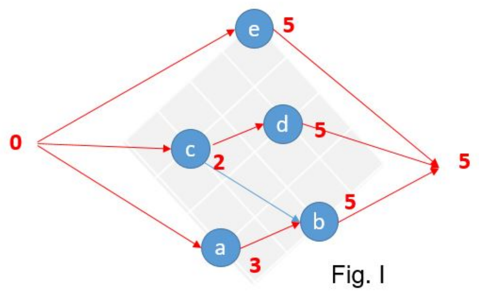


Fig. I

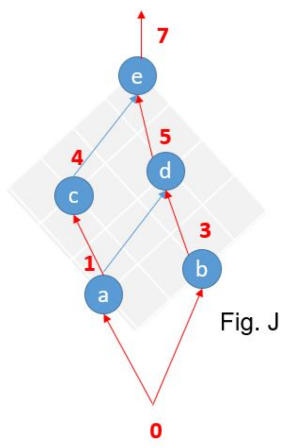


Fig. J

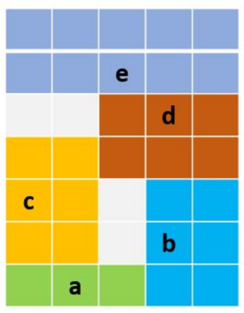
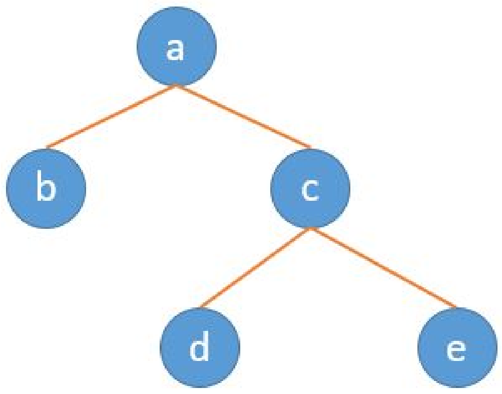


Fig. K

We traverse through the horizontal and vertical constraint graph and calculate the longest path for each node, shown in Fig.I and Fig.J. Therfore, the minimum area cost is 35 (5×7). The compact packing is shown in Fig.K.

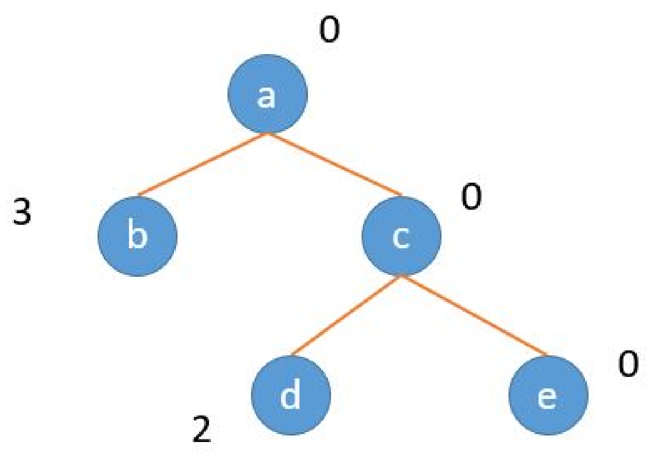
d.

The compact floorplan of Fig.4 is in Fig.K, we can easily form its corresponding B*tree (the tree transformed from a floorplan may not be unique in general case):



e.

First, calculate the x-coordinate by simply traversing over the tree. The x-coordinate of each node is equal to the x-coordinate of its parent plus the width of its parent, if it is a left child, otherwise, its x-coordinate is the same as its parent. (Shown in the figure below)



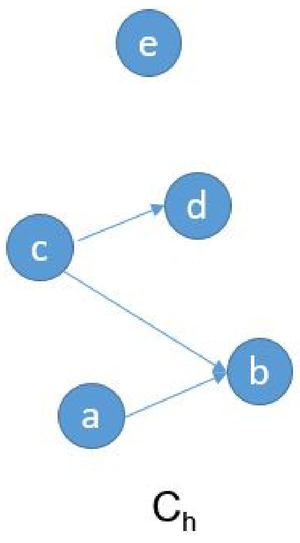
Then we can calculate the contour to get the y-coordinate of each block.
Based on the x-coordinate calculated in the previous step, we can formulate the interval of each block. We maintain the contour by a double link-list, which is initialize by $(0,0) \to (\infty,0)$. Everytime when we add a new block into the contour, we find nodes in the linked-list that are in the interval of that block, and find the maximum y-coordinate in these node, then delete these nodes and add the intervals of this block into the linked-list with y-coordinate equal to sum of the maximum y calculated previously and the block height.
The step by step modification of the contour is shown in the table below.

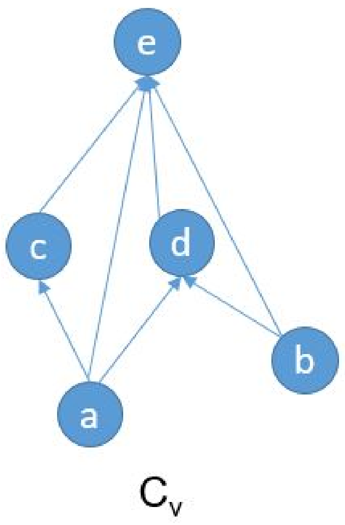
Step	**Current contour**	**new add block (interval$\times$ H)**	**y-coordinate**
1	$(0,0) \to (\infty,0)$	$a:(0,3)\times 1$	0
2	$(0,0) \to (0,1) \to (3,1) \to (3,0) \to (\infty,0)$	$b:(3,5)\times 3$	0
3	$(0,0) \to (0,1) \to (3,1) \to (3,3) \to (5,3) \to (5,0) \to (\infty,0)$	$c:(0,2)\times 3$	1
4	$(0,0) \to (0,4) \to (2,4) \to (2,1) \to (3,1) \to (3,3) \to (5,3) \to (5,0) \to (\infty,0)$	$d:(2,5)\times 2$	3
5	$(0,0) \to (0,4) \to (2,4) \to (2,5) \to (5,5) \to (5,0) \to (\infty,0)$	$e:(0,5)\times 2$	5
6	$(0,0) \to (0,7) \to (5,7) \to (5,0) \to (\infty,0)$		

Finally, the coordinate of each block is as follows:

	a	b	c	d	e
x	0	3	0	2	0
y	0	0	1	3	5

f.
The horizontal and vertical constrainst graph of TCG pair is shown in the figure below.

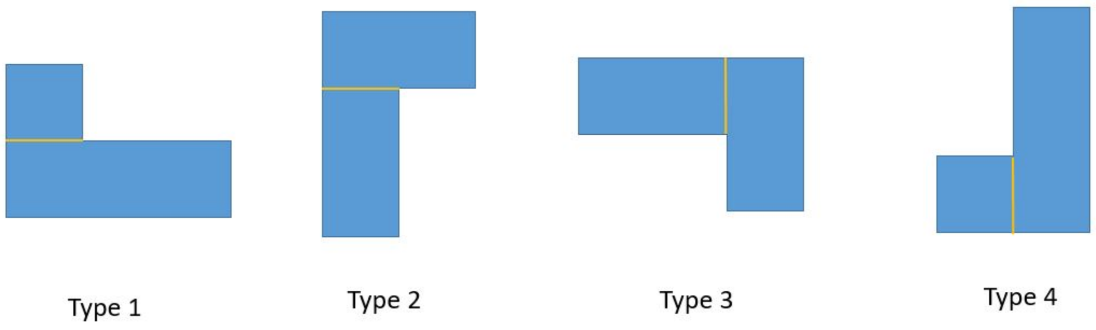




Problem 5 [Collaborator: None]

[Ref. Wu, Chang, Chang, “Rectilinear block placement using B*-trees,” ACM TODAES, 2003 (ICCD-01)]

a.
For a L-shape block, there are 4 types of orientation, shown in the figure below.
If the orientation of L-shape block is of type 1 and 2, we horizontally cut it into two blocks, and let the block in the top be the right child of the bottom one, these two nodes are called LC pairs and cannot be seperate when perturbing the tree.
On the other hand, if the orientation is of type 3 and 4, we vertically cut it into two blocks, and let the block in the right be the left child of the left one, and simlilarly it is a LC pair, cannot be serperate when perturbing the tree.



Therefore, the B*-tree of this placement is as follows (Fig.B), note that the representation is not unique:

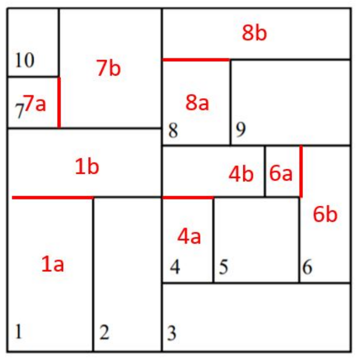


Fig. A

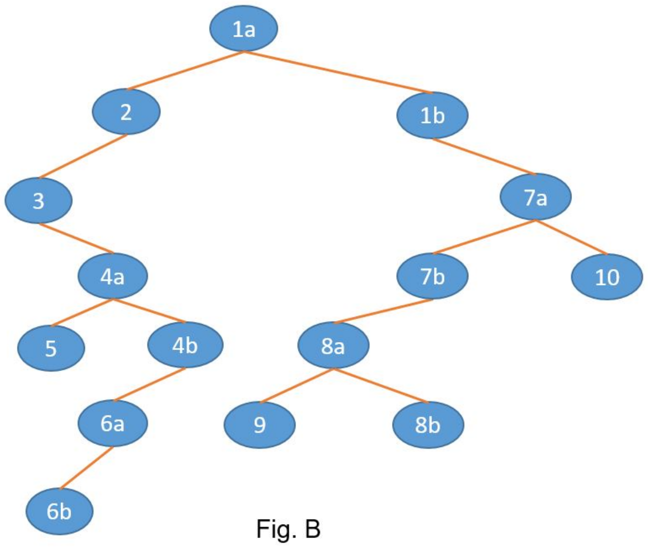


Fig. B

b.

Step1.

First, calculate the x-coordinate by simply traversing over the tree. The x-coordinate of each node is equal to the x-coordinate of its parent plus the width of its parent, if it is a left child, otherwise, its x-coordinate is the same as its parent.

Step2. (Refer to the figure below)

Then we can calculate the contour to get the y-coordinate of each block.

Based on the x-coordinate calculated in the previous step, we can formulate the interval of each block. We maintain the contour by a double link-list, which is initialize by $(0, 0) \rightarrow (\infty, 0)$. Everytime when we add a new block into the contour, we find nodes in the linked-list that are in the interval of that block, and find the maximum y-coordinate in these node, then delete these nodes and add the intervals of this block into the linked-list with y-coordinate equal to sum of the maximum y calculated previously and the block height.

Notice that if a type 3 and 4 LC pairs occurs, the y contour of the two block might be modify to the greater one!

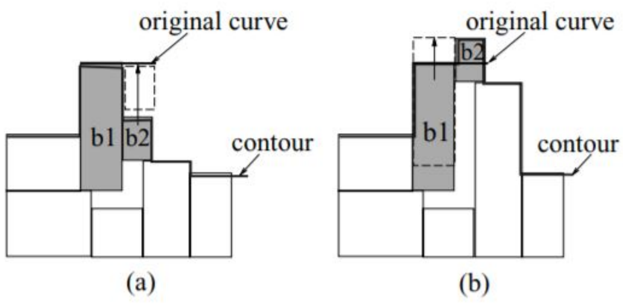
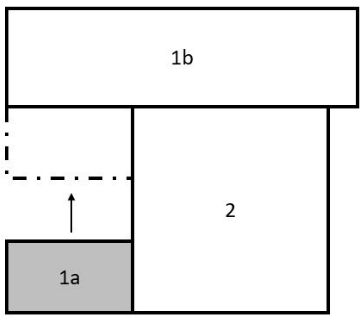


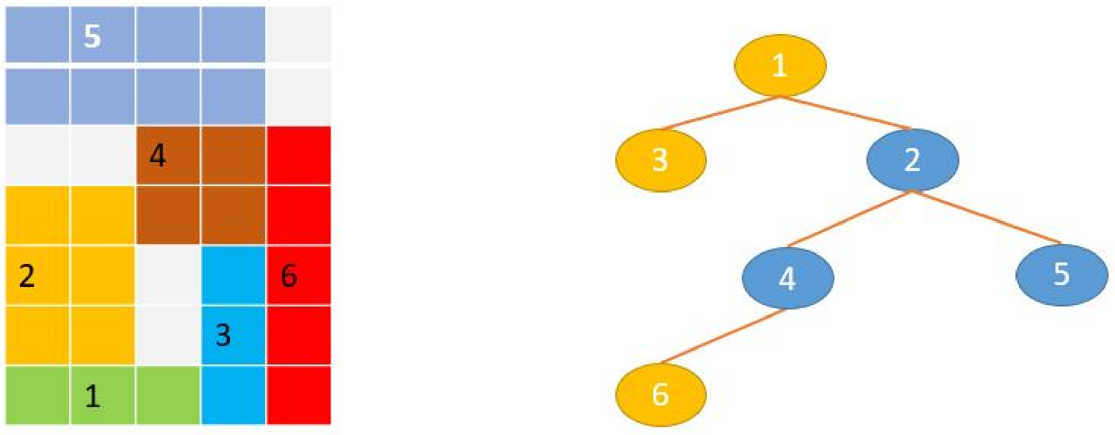
Fig. 7. Placing two subblocks b_1 and b_2 of an L-shaped block: (a) if the contour is lower than the top profile sequence at b_2, then we pull b_2 up to meet the top profile sequence; (b) if the contour is higher than the top profile sequence at b_2, then we pull b_1 up to meet the top profile sequence.

Step.3 (Refer to the figure below)

For all LC pairs with type 2 orientation, increase the y-coordinate of the bottom block, so that the two block is connected.



c.
The modules of the bottom boundary of the floorplan is the left skew of the tree: 1a, 2, 3 in this case.
However, this is not always true, refer to the figure below, we still need to packed the blocks and see which blocks has 0 x values.



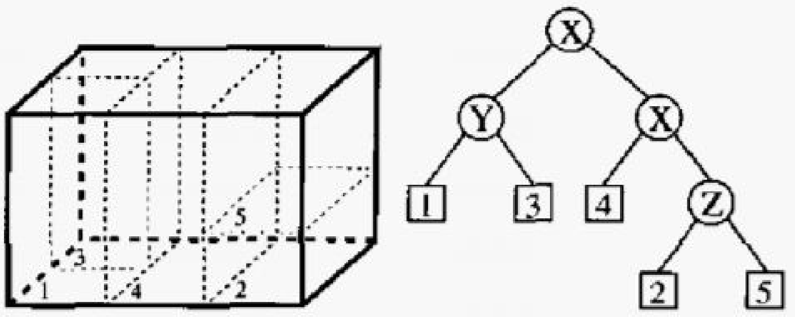
Notice that if the node is one of the LC pairs with type 2, 3, 4 orientation, it may not be on the bottom boundary, we need to check its y coordinate.
The modules of the left boundary of the floorplan is the right skew of the tree: 1a, 1b, 7a, 10 in this case.

Problem 6 [Collaborator: None]

	Strengths	**weaknesses**
Λ-shaped	The clutersing stage consider area utilization and module local connectivity, therefore, it can have more compact floorplan.	It does not have the view of the global configuration at the earlier stages, therefore, the global wire length is not well optimized.
V-shaped	It consider the global interconnect at the earilier stages, therefore, it can have better utilization on global information.	During partition stage, the block will be placed in a fixed outline, therefore, its area will not be as compact as Λ-shaped methods.

Problem 7 [Collaborator: None]

a. Slicing tree
[Ref. “Floorplanning for 3-0 VLSI Design”, Lei Cheng, Liang Deng, Martin D.F. Wong (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5545492)]



(a) Slicing structure (b) Slicing tree

Figure 1: 3-D slicing floorplan

<Representation>
A slicing floorplan can be represented by an oriented rooted binary tree, called a slicing tree (see Fig. l(b)). Each internal node of the tree is labeled by X, Y, or Z. The label X means that the corresponding supermodule is cut by a plane that is perpendicular to the z axis. The same is true for label Y and label Zin terms of the y axis and z axis. respectively.
We represnet the slicing tree same as the classical slicing tree Polish expression representation.

<Algorithm>
We define the same operation as the classical slicing tree one, and running SA algorithm.

b. B*-tree
[Ref. "Temporal floorplanning using the T-tree formulation", Ping-Hung Yuh, Chia-Lin Yang, Yao-Wen Chang]

<Represnetation>
Each node have 3 children now. The left child must be placed adjacent to the parent block with in the z direction, the middle child must be placed in the y direction while z is the same as the parent block, the right child is placed in the x direction with z, y coordinate same as the parent

<Algorithm - from T-tree to 3D-packing>
Step1.
DFS and find the z coordinate.
Step2.
DFS the T-tree and if there is a right child split it(see Fig. 4). After this operation there will be several B*-trees. Applying the contour data structure to determine the y coordinate of each block. Notice that the step 1 and step 2 is simliar to 2D B*tree methods, if we project the 3D block to the yz-plane.
Step3.
When maintaining the contour, we maintain a list with all task whose y, z are already determined, and if the projection of the new task on the yz plane is overlap, we choose the max x in the list.

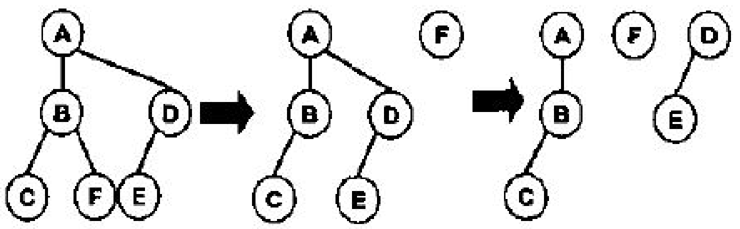


Figure 4: The T-tree decomposition process

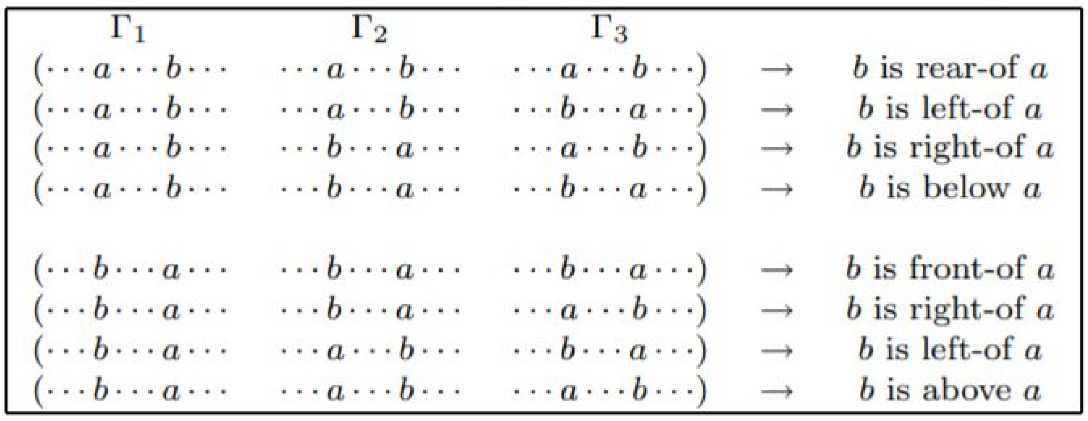
Intuitively, this algorithm can be see as packing many B*trees in the x direction, and the y, z coordinate of the root is determine by the place it is placed in the T-tree.

<Algorithm>
Besides the rotation operation (the rotation has 6 directions, not include flipping), the other operation we choose to be the same. Also, we apply SA algorithm to solve this problem.

c. Sequence pair
[Ref. "The 3D-Packing by Meta Data Structure and Packing Heuristics", Hiroyuki Yamazaki, keishi sakanushi, shigetoshi nakatake, yoji kajitani (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.7728&rep=rep1&type=pdf)]

<Representation>
The seq-pair is extended to sequence triple $(\Gamma_1, \Gamma_2, \Gamma_3)$, and the decoding rule of the triplets is hown in the figure below.

Table 1 Decode rule of the seq-triple sorted w.r.t. the 1st sequence.



Γ_1	Γ_2	Γ_3		
$(\cdots a \cdots b \cdots$	$\cdots a \cdots b \cdots$	$\cdots a \cdots b \cdots)$	$\rightarrow$	b is rear-of a
$(\cdots a \cdots b \cdots$	$\cdots a \cdots b \cdots$	$\cdots b \cdots a \cdots)$	$\rightarrow$	b is left-of a
$(\cdots a \cdots b \cdots$	$\cdots b \cdots a \cdots$	$\cdots a \cdots b \cdots)$	$\rightarrow$	b is right-of a
$(\cdots a \cdots b \cdots$	$\cdots b \cdots a \cdots$	$\cdots b \cdots a \cdots)$	$\rightarrow$	b is below a
$(\cdots b \cdots a \cdots$	$\cdots b \cdots a \cdots$	$\cdots b \cdots a \cdots)$	$\rightarrow$	b is front-of a
$(\cdots b \cdots a \cdots$	$\cdots b \cdots a \cdots$	$\cdots a \cdots b \cdots)$	$\rightarrow$	b is right-of a
$(\cdots b \cdots a \cdots$	$\cdots a \cdots b \cdots$	$\cdots b \cdots a \cdots)$	$\rightarrow$	b is left-of a
$(\cdots b \cdots a \cdots$	$\cdots a \cdots b \cdots$	$\cdots a \cdots b \cdots)$	$\rightarrow$	b is above a

Based on the RL, FR, AB topology, we can decode each pair of them based on the sequence triple intp three constraint graphs G_{RL}, G_{FR}, G_{AB}, then the longest path length to each vertex locates the corresponding box.
However, any 3D-packing by the seq-triple is tractable(there is a linear order for all the boxes that the box is extracted one by one in above-front wihtout moving the remain box). For example, the 3D-packing in Fig.6 is intractable and it cannot be represent by seq-triple. Moreover, even tractable packing may not be represent by seq-triple, as Fig. 5.

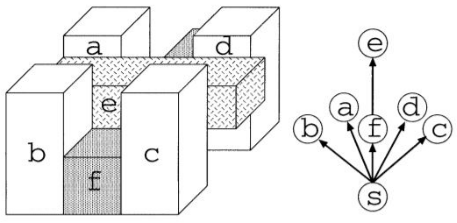


Fig. 5 A 3D-packing whose topology is not covered by the seq-triple, while it is by seq-quintuple (*ebafcd*, *abfdce*, *fcbeda*, *bceadf*, *abcdfe*).

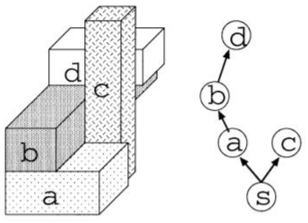


Fig. 6 Another example whose topology corresponds to no seq-triple, while to seq-quintuple (*adbc*, *bcda*, *acdb*, *bacd*, *acbd*).

Therefore a sequence quintuple $(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5)$ is needed.

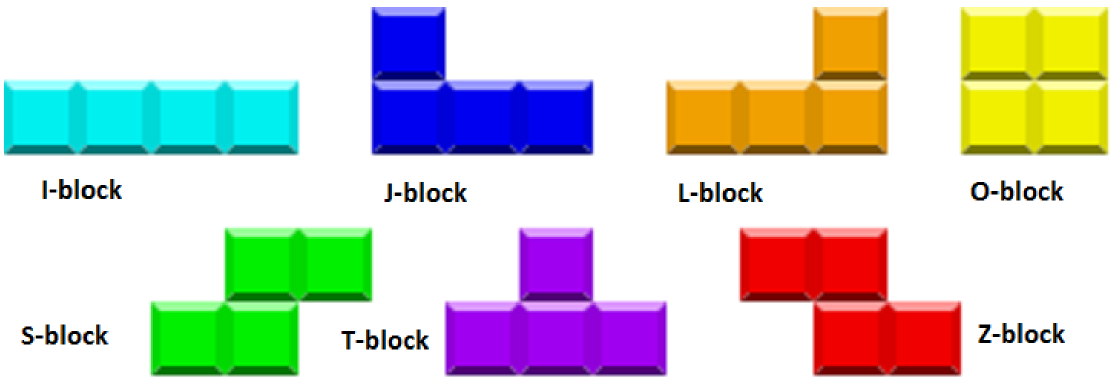
<Algorithm - from seq-quintple ot 3D-packing>
Step1.
Construct right-left constraint graph based on Γ_1, Γ_2, it follows the same definition as for the seq-pair but restricted only to the right-left relation.
Also, construct the front-rear constraint graph from Γ_3, Γ_4, it follows the same definition as for the seq-pair but restricted only to the front-rear relation
Step2.
Find the longest path on the graph G_{RL} and G_{FR} and let it be the x,y of the box.
Step3.
Construct the above-below constraint graph G_{AB} as follows:
For each box a, b, and edge from a to b is added iff
(1) a, b are xy-overlapping and
(2) $\Gamma_5^{-1}(a) < \Gamma_5^{-1}(b)$
Step4.
Determine the longest path on G_{AB}
The origianl has proved that:
the above algorithm provides a unique 3D packing, and
there must be a seq-quintuple for any 3D-packing

<Algorithm>
Operation same as the classical SP, and apply SA algorithms.

Problem 8 [Collaborator: B05901053 吳宥璁]

Introduction

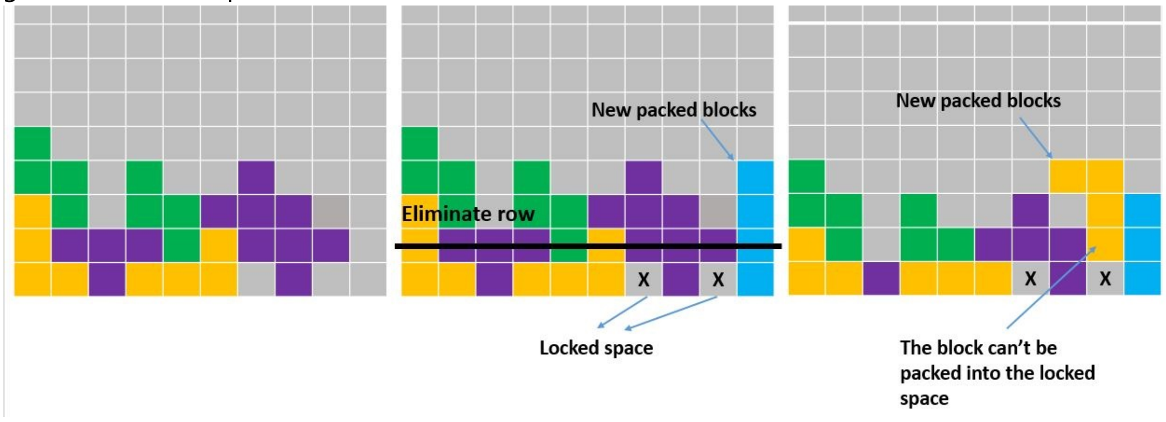
A well known game called Tetris battle aims to packed 7 kinds of blocks (see figure below), which is given sequantially, into a fixed width box with no overlaps. Also, if a row is fulled with block, you can eliminate the row and get scores. The scoring function is shown in the problem abstraction part.



The interface of the game is as following:

The game ends if your packed block height is higer than the box height.

Problem Restriction
Since this problem may be a bit complex, we make some restriction on this problem:
(1) If We eliminate a row and there are rows that are not eliminate under it, we cannot filled the blank space and eliminate these rows, the blank spaces are locked! (see figure below)
(2) To prevent non-terminate games, assumes that constant large amount of blocks is given sequqntially, and the game ends if the packed block height exceeds the box height or “the given blocks is all packed”.



Problem Abstraction
Given a block vectors, ex. $\{T, I, J, L, O, O, Z, S, S, T, J, \ldots, T\}$.
Notice that the orders of blocks is important, for example, if block vector $\{T, O\}$ is given, then O block will not be under the T block, since the block in the game is given sequntially. We called this a **ordering constraint** in this problem.

And optimization terms is to packed these blocks, while satisfying **ordering constraints**, to get the higher scores.

The scoring function is define here:
(1) If you packed to the height of the box height, the games end. You get the scores by the accumulated scores calculated in (2)~(3).
(2) If you continiuisly eliminate rows, that is there are no other block packings between these two adjacency row elimination, we called it a **combo**. And table of scores to number of

elimination rows and number of combos is shown below.
(3) If there are no non 4-rows eliminate between two 4-rows elimination (it must be down by packing I blocks), then the second 4-rows elimination gets 6 points rather than 4.

Height	COMBO	LS	ΣLS
1		0	0
2	1	1	1
3	2	1	2
4	3	2	4
5	4	2	6
6	5	3	9
7	6	3	12
8	7	4	16
9	8	4	20
10	9	4	24
11	10	4	28
12	11	4	32
13	12	4	36
14	13	4	40
15	14	4	44
16	15	4	48
17	16	4	52
18	17	4	56
19	18	4	60
20	19	4	64
	20	4	68

Algorithms for solving the problem

(1) Heuritics: a compacted packing at any times may get higher scores

We aimed to solve this problem by a B*-tree structure, but B*-tree can't represent non-compacted packings. For instance, if there are three O-type blocks called A, B, C, we can't represent A, C, B since if we packed A, B but not C yet, the packing is not compacted.

However, by heuristic, compacted packed structure can get higher scores in this problem.

(2) The ordering constriants must be satisfy in B*tree

If a block A has smaller index in vector than B, we called A preceeds B.

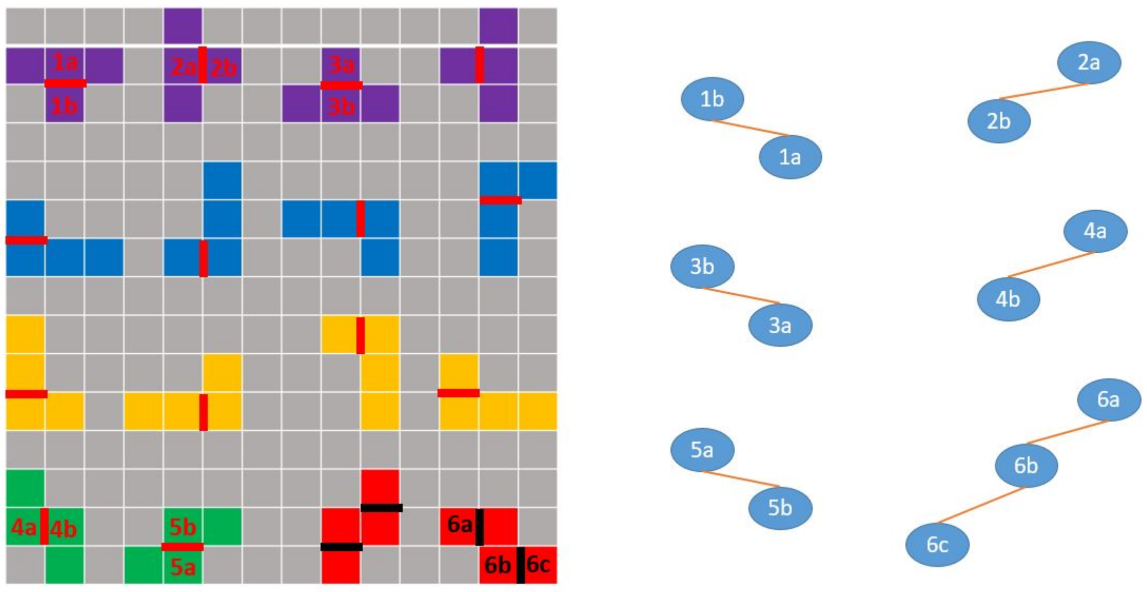
For any parent-children pairs, **the parent block must preceeds children blocks**; otherwise, the sequqntially given constraint might not be satisfy.

(3) Methods for representing 7 kinds of blocks in the B*tree

Similar to problem 5 in this homework, we divide a non-rectangle block into several rect blocks and formed them into LC-pairs.

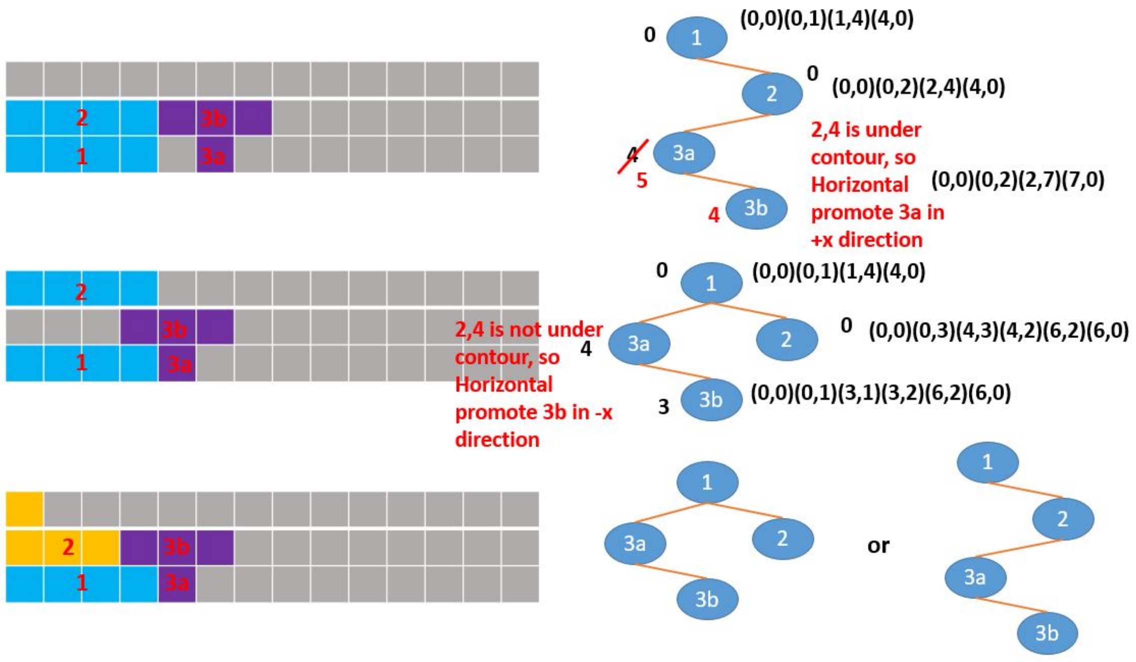
Notice that similar to problem 5, if a rectlinear has a blank space in the right and bottom side, we should cut it horizontally; otherwise, if a single block is packed in that blank space, there will be two blocks that is adjacent to the that block, which only one left child is present in a B*tree.

The divided rectilinear and LC pairs of some complex types is shown in the figure below.



Notice that there are only one type of cuts on a single type. If a type is in vertical cut, then we apply the procedure as problem 5, when maintaining the contour data structure, we promote the LC pairs to satisfy the contour of itself. If a type is a horizontal cut, it is similar to the horizontal cut in problem 5 of L-shape. But their is a bit different in T-shape or S-shape blocks. We must promote these block in horizontal direction to satisfy its contour. Examples are shown below for T-shape.

The single number indicates the x coordinate, and the linked-list structure of the contour is shown also for each hypernode. Notice that in our method contour depends on previous x, and x value depends on previous contour, so we need to update x and contour per block. Also, the block traverse order follows: parent > left > right.



(4) Apply simulated annealing algorithms

We'll discuss the four basic ingredients of a SA algorithms:

The **cost function** is defined below, where the game_score term corresponds to the score of the problem

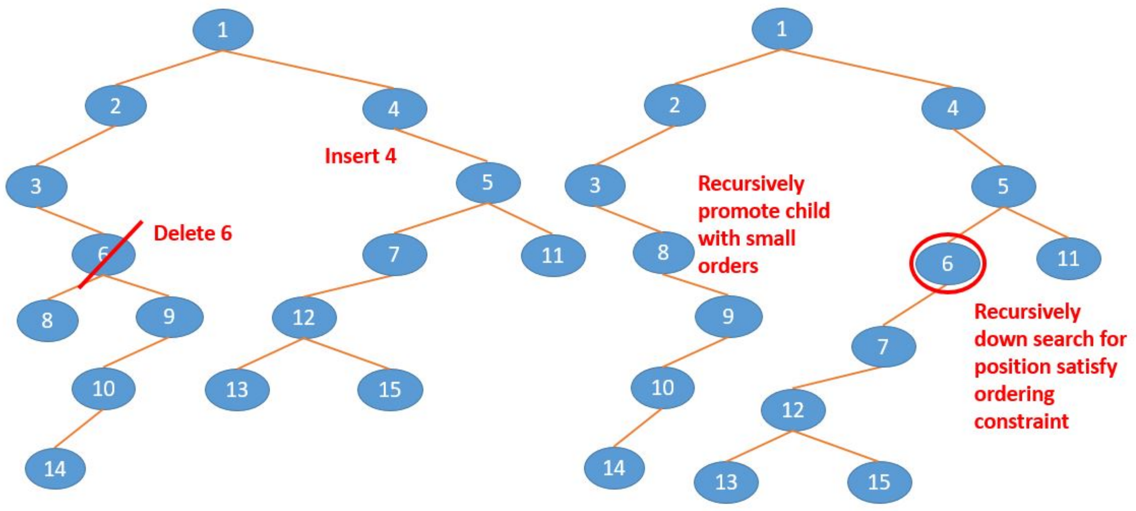
$cost = \alpha \times game\_score + (1 - \alpha) \times abs(y_{packed} - y_{box})$

```
combo_count = 0
accumulate_score = 0
previous_eliminate_rows = 0
while Travering the B*tree
    update x as presented earilier
    update contour as presented earilier
    if contour exceeds box height
        break
    if contour acheive width of the box:
        // compare current contour and previous contour to check to
        // get the number of elimnate rows
        num_eliminate_rows = get_eliminate_rows(current contour, previous contour)
        if num_eliminate_rows == 0
            combo_count = 0
        else if num_eliminate_rows != 0
            combo_count += 1
            // get score from the scoring table, and accumulate
            accumulate_score += get_score(combo_count, num_eliminate_rows)
        else if previous_eliminate_rows == 4
            accumulate_score += 6

        previous_eliminate_rows = num_eliminate_rows
```

We define **3 types of perturbing opertations** on the tree:

1. Rotate blocks: rotate the blocks direction, some block has 4 direction while some has 2 or only 1 direction.
2. Delete and insert nodes: Choose two node, delete it from tree and insert as the other nodes child. Notice that the allowed operation is that the parent order must exceeds its child. See the following example.



Insert 4
Delete 6
Recursively promote child with small orders
Recursively down search for position satisfy ordering constraint

3. Swap subtrees: Select one nodes, swap its two subtrees. (the ordering constraint will always be satisfied)

The **annealing scheduling** needs to be tuned by experiments.

The **solution space** is all compacted packing and satisfied with ordering constraints and the fixed y width constraint, and this may get high score by heuristic as mentioned earlier.

Also the **initial solution** can be any B*tree representations that satisfy ordering constrainst.