HW2

Saturday, 5 April 2025 9:07 PM

1. (30 pts) Given the following Polish expression, E = 12H3V4HV5.
(a) (6 pts) Dose the above expression satisfy the balloting property? Justify your answer.

$e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7\ e_8\ e_9$
E = 1 2 H 3 V 4 H V 5

No of operands = 1 2 2 3 3 4 4 4 5
No of operators = 0 0 1 1 2 2 3 4 4

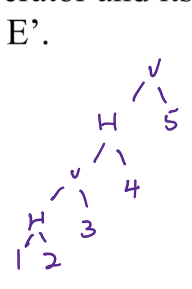
It does not satisfy the balloting property, because exists # of operands = # of operators for subexpression e_8

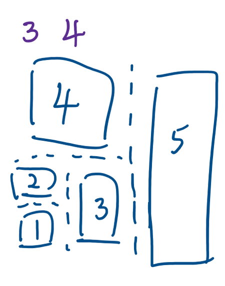
(b) (6 pts)
Is E a normalized Polish expression? If not, exchange an operator and its adjacent operand to transform E into a normalized Polish expression E'.

No, E is not even a Polish expression.

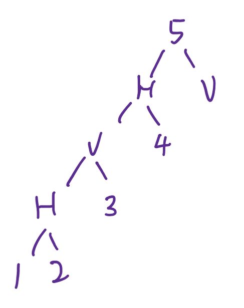
swap V and 5

Do M3 to E ⇒ E = 1 2 H 3 V 4 H 5 V
# of operands = 1 2 2 3 3 4 4 5 5
# of operators = 0 0 1 1 2 2 3 3 4



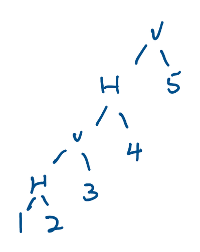


(c) (6 pts)
Give the slicing tree corresponding to the expression E, or explain why no such a slicing tree exists. Also, give the slicing tree corresponding to the "resulting" normalized Polish expression E' , if E is not a normalized Polish expression.



this slicing tree should not exists, because the root is operand, and operand only can be a leaf

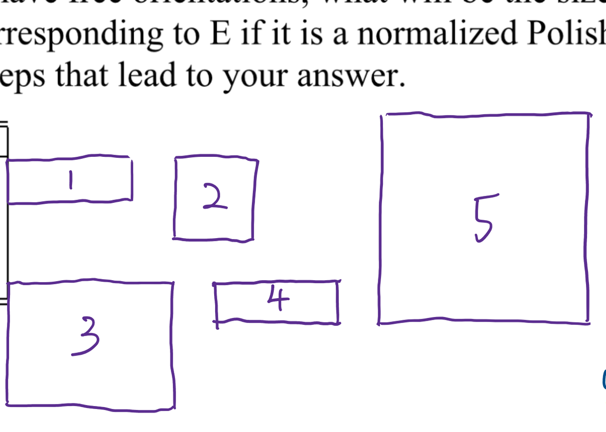
E' = 1 2 H 3 V 4 H 5 V

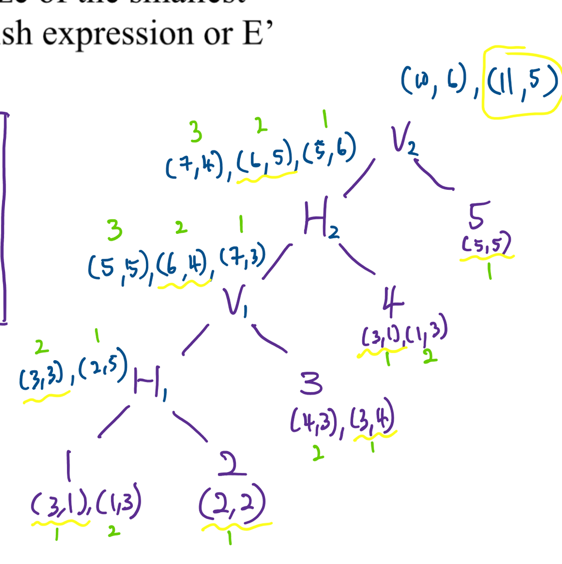


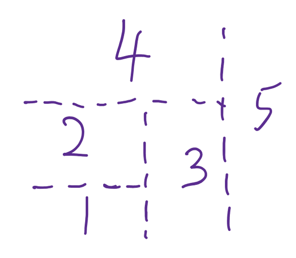
(d) (12 pts)
Assume the modules 1, 2, ..., 5 have the sizes and shapes indicated in Table 1. If all modules are hard and have free orientations, what will be the size of the smallest bounding rectangle corresponding to E if it is a normalized Polish expression or E' otherwise? Show all steps that lead to your answer.

Module No.	Width	Height
1	3	1
2	2	2
3	4	3
4	3	1
5	5	5

Figure 1: Table 1.



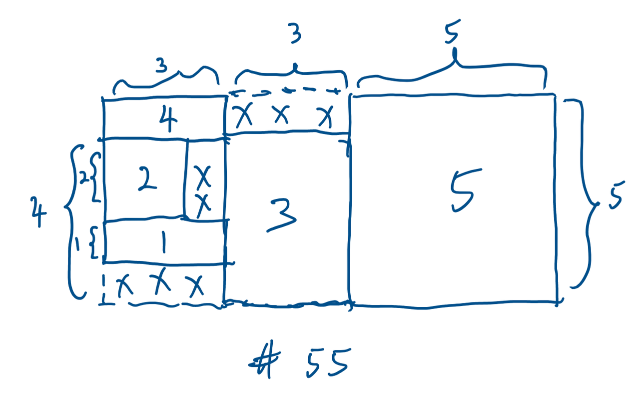




1.) Get all possible shape to all operands
2.) Check operands 1, 2 and get shape of H_1 = { (3,3), (2,5) }
3.) Check H_1, 3 and get shape of V_1 = { (5,5), (6,4), (7,3) }
4.) Check V_1, 4 and get shape of H_2 = { (7,4), (6,5), (5,6) }
5.) Check H_2, 5 and get shape of V_2 = { (10,6), (11,5) }
6.) Chose from smallest shape (11, 5)

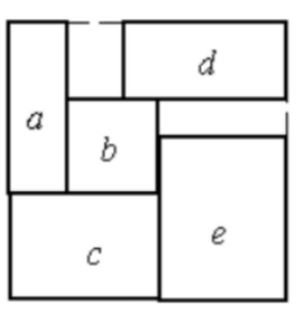
※ green color is order of sorting use stokmeyer to do
(Under H, decreasing width order)
(If left have larger width, then consider next shape in left, vice versa)

※ Yellow is the shape origin of smallest Area.



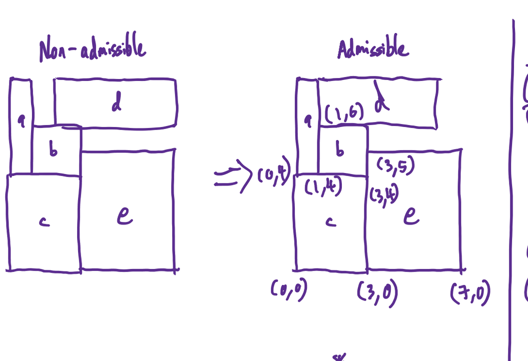
55

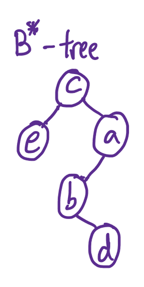
2. (20 pts)
Derive the B*-tree for the packing of the five modules, a, b, c, d, and e shown in Figure 2. Show all steps for computing the coordinates of the modules from the resulting B*-tree?



Module	Width	Height
a	1	4
b	2	2
c	3	4
d	5	2
e	4	5

Figure 2: A packing for the five modules a, b, c, d, and e.





Modules	Coordinates
(0,0) c root	(0,0) root of the B*-tree Left child → e : x = 0 + 3 = 3, y = 0 ⇒ (3,0) Right child → a : x = 0, y = 0 + 4 = 4 ⇒ (0,4)
(3,0) e	No child
(0,4) a	Left child → b : x = 0 + 1 = 1, y = 4 ⇒ (1,4)
(1,4) b	Right child → d : x = 1, y = 4 + 2 = 6 ⇒ (1,6)
(1,6) d	No child

3. (25 pts)
For the floorplanning problem discussed, we were asked to minimize the area of the final bounding rectangle for the given set of hard blocks. Suppose we need to pack a set of hard blocks into a die with a fixed bounding rectangle (i.e., fixed-outline floorplanning). Discuss how you can modify the cost function of floorplanning to address the fixed-outline floorplanning.
(Note: in a slide, we showed a cost function is like

Cost Function

- $\phi = A + \lambda W$
  - A: area of the smallest rectangle
  - W: overall wiring length
  - λ : user-specified parameter

How to enhance?)

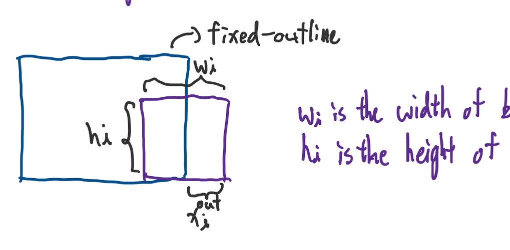
Add a penalty P into cost function to restrict the block not to excess the outline.

$\phi = \beta A + \lambda W + \alpha \cdot P_{boundary}$

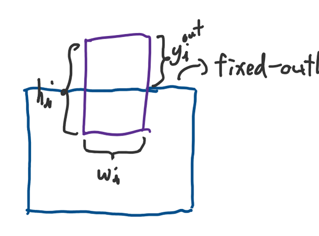
※ $P_{boundary} = \sum_i \left(\max(0, x_i^{out}) \cdot h_i + \max(0, y_i^{out}) \cdot w_i \right)$

x_i^{out} is the width of block i that excess fixed-outline
y_i^{out} is the height of block i that excess fixed-outline

ex:

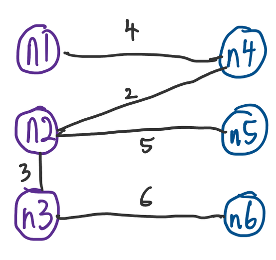


w_i is the width of block i
h_i is the height of block i



β is user-specified parameter, default = 0
α is user-specified parameter
$P_{boundary}$ is total area that excess fixed-outline

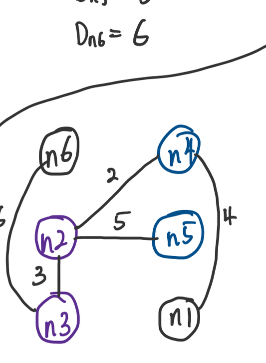
4. (25 pts) Partitioning
Let the three tuple (u, v, w) denotes an edge (u, v) with weight w. Given a circuit C of 6 vertices, n1, n2, ...,n6, and 5 edges, C = {(n1, n4, 4), (n4, n2, 2), (n2, n3, 3), (n2, n5, 5), (n3, n6, 6)}, apply the Kernighan-Lin heuristic to find the balanced min-cut for C with the initial partitions {n1, n2, n3} and {n4, n5, n6}. Show all steps that lead to your answer.



Iteration 1:
$I_{n1} = 0$, $E_{n1} = 4$ — $D_{n1} = E_{n1} - I_{n1} = 4$
$I_{n2} = 3$, $E_{n2} = 2+5 = 7$ — $D_{n2} = 7 - 3 = 4$
$I_{n3} = 3$, $E_{n3} = 6$ — $D_{n3} = 6 - 3 = 3$
$I_{n4} = 0$, $E_{n4} = 4+2 = 6$ — $D_{n4} = 6$
$I_{n5} = 0$, $E_{n5} = 5$ — $D_{n5} = 5$
$I_{n6} = 0$, $E_{n6} = 6$ — $D_{n6} = 6$

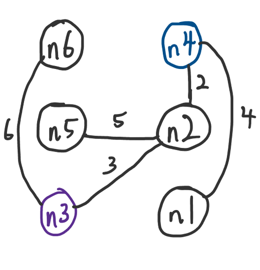
$g_{14} = D_1 + D_4 - 2 \cdot C_{14} = 4 + 6 - 2 \cdot 4 = 2$
$g_{15} = 4 + 5 - 2 \cdot 0 = 9$
$g_{16} = 4 + 6 - 2 \cdot 0 = 10$ (maximum) ($\hat{g}_1$)
$g_{24} = 4 + 6 - 2 \cdot 2 = 6$
$g_{25} = 4 + 5 - 2 \cdot 5 = -1$
$g_{26} = 4 + 6 - 2 \cdot 0 = 10$
$g_{34} = 3 + 6 - 2 \cdot 0 = 9$
$g_{35} = 3 + 5 - 2 \cdot 0 = 8$
$g_{36} = 3 + 6 - 2 \cdot 6 = -3$

(1 ⟷ 6)



$D_2' = D_2 + 2 \cdot C_{21} - 2 \cdot C_{26} = 4 + 2 \cdot 0 - 2 \cdot 0 = 4$
$D_3' = 3 + 2 \cdot 0 - 2 \cdot 6 = -9$
$D_4' = 6 + 2 \cdot 0 - 2 \cdot 4 = -2$
$D_5' = 5 + 2 \cdot 0 - 2 \cdot 0 = 5$

$g_{24} = 4 + (-2) - 2 \cdot 2 = -2$
$g_{25} = 4 + 5 - 2 \cdot 5 = -1$ (maximum) swap n2 and n5 ($\hat{g}_2$)
$g_{34} = -9 + (-2) - 2 \cdot 0 = -11$
$g_{35} = -9 + 5 - 2 \cdot 0 = -4$



(2 ⟷ 5)

$D_3'' = -9 + 2 \cdot 3 - 2 \cdot 0 = -3$
$D_4'' = -2 + 2 \cdot 0 - 2 \cdot 2 = -6$

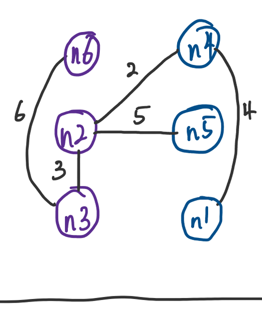
$g_{34} = -3 - 6 - 2 \cdot 0 = -9$ (maximum)

swap n3 and n4 ($\hat{g}_3 = -9$)

Summary iteration 1: $\hat{g}_1 = 10$, $\hat{g}_2 = -1$, $\hat{g}_3 = -9$

Largest partial sum $\max \sum_{i=1}^{k} \hat{g}_i = 10$ (k = 1) ⇒ swap n_1 ⟷ n6

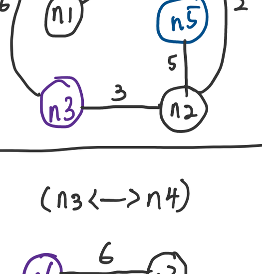
Iteration 2:



$D_1 = -4$
$D_2 = 4$
$D_3 = -9$
$D_4 = -2$
$D_5 = 5$
$D_6 = -6$

$g_{12} = -4 + 4 - 2 \cdot 0 = 0$ (maximum) ($\hat{g}_1 = 0$)
$g_{13} = -4 + (-9) - 2 \cdot 0 = -13$
$g_{16} = -4 - 6 + 2 \cdot 0 = -10$
$g_{42} = -2 + 4 - 2 \cdot 2 = -2$
$g_{43} = -2 - 9 - 2 \cdot 0 = -11$
$g_{46} = -2 - 6 - 2 \cdot 0 = -8$
$g_{52} = 5 + 4 - 2 \cdot 5 = -1$
$g_{53} = 5 - 9 - 2 \cdot 0 = -4$
$g_{56} = 5 - 6 - 2 \cdot 0 = -1$

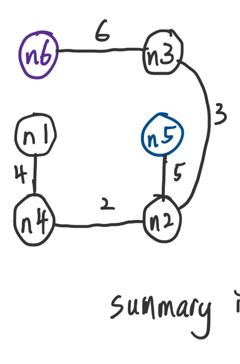
(n1 ⟷ n2)



$D_3' = -9 + 2 \cdot 3 - 2 \cdot 0 = -3$
$D_4' = -2 + 2 \cdot 4 - 2 \cdot 2 = 2$
$D_5' = 5 + 2 \cdot 0 - 2 \cdot 5 = -5$
$D_6' = -6 + 2 \cdot 0 - 2 \cdot 0 = -6$

$g_{34} = -3 + 2 - 2 \cdot 0 = -1$ (maximum) ($\hat{g}_2 = -1$)
$g_{35} = -3 - 5 - 2 \cdot 0 = -8$
$g_{64} = -6 + 2 - 2 \cdot 0 = -4$
$g_{65} = -6 - 5 - 2 \cdot 0 = -11$

(n3 ⟷ n4)



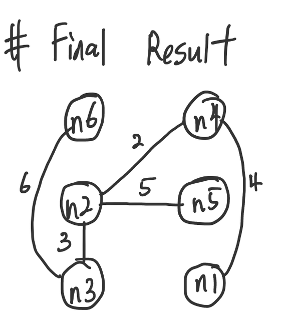
$D_5'' = -5 + 2 \cdot 0 - 2 \cdot 0 = -5$
$D_6'' = -6 + 2 \cdot 6 - 2 \cdot 0 = 6$

$g_{56} = -5 + 6 - 2 \cdot 0 = 1$ (maximum) ($\hat{g}_3 = 1$)

summary iteration 2: $\hat{g}_1 = 0$, $\hat{g}_2 = -1$, $\hat{g}_3 = 1$

Largest partial sum $\max \sum_{i=1}^{k} \hat{g}_i = 0$ (k = 1, 3) ⇒ Stop

Final Result



Cut size = 7