# CS 601 - Advanced Analysis of Algorithms

1. $$g_1(n) = 2^{\sqrt{\log_2 n}}$$

$$g_2(n) = n^{4/3}$$

$$g_3(n) = n^{\log_2 n}$$

$$g_4(n) = n(\log_2 n)^3$$

$$\log_2 g_1(n) = \log_2 2^{\sqrt{\log_2 n}} = \sqrt{\log_2 n} = \sqrt{z} \quad \text{where } z = \log_2 n$$

$$\log_2 g_2(n) = \log_2 n^{4/3} = \frac{4}{3}\log_2 n = \frac{4}{3} z$$

$$\log_2 g_3(n) = \log_2 n^{\log_2 n} = (\log_2 n)^2 = z^2$$

$$\log_2 g_4(n) = \log_2\left(n(\log_2 n)^3\right) = \log_2 n + \log_2(\log_2 n)^3 = z + \log_2 z^3 = z + 3\log_2 z$$

Now, since square root grows slower than degree one polynomials, which in turn grows slower than degree 2 polynomials, we can write

$$\overset{g_1}{\sqrt{z}} < \overset{g_2}{\frac{4}{3} z} < \overset{g_3}{z^2}$$

Clearly, $\log_2 g_4(n) = z + 3\log_2 z$ performs / runs longer than $\log_2 g_1$. So, $g_4 > g_1$

also $\log_2 g_4(n) = z + 3\log_2 z \overset{?}{>} \frac{4}{3} z = \log_2 g_2(n)$

$$\lim_{z\to\infty} \frac{z + 3\log_2 z}{\frac{4}{3} z} = \lim_{z\to\infty} \frac{3}{4} + \frac{9}{4z}\log_2 z$$

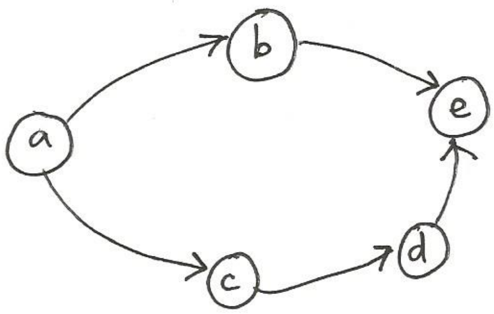
$$= \frac{3}{4} < 1$$

So, $g_4 < g_2$ and $g_4 > g_1$

∴ we can write

$$\boxed{\begin{array}{c} 2^{\sqrt{\log_2 n}} < n(\log_2 n)^3 < n^{4/3} < n^{\log_2 n} \\ g_1 < g_4 < g_2 < g_3 \end{array}}$$

2.




Three Topological orderings:

a, b, c, d, e

a, c, b, d, e

a, c, d, b, e

3. True One argument is as follows: $e^*$ is the first edge that would be considered by Kruskal's algorithm, and so it will be included in the minimum spanning tree.

4. $p_0 = 0.4$ Let the event $S = s_k$ denote the emission of symbol $s_k$ by the source.

$p_1 = 0.3$

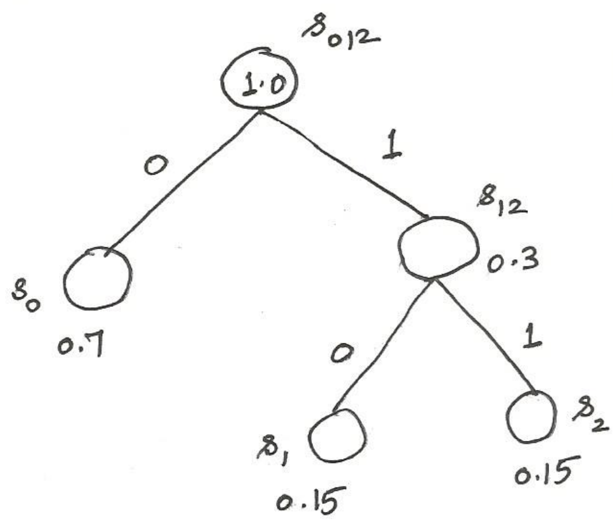
$p_2 = 0.2$

$p_3 = 0.1$

Hence, $I(s_k) = \log_2\left(\frac{1}{p_k}\right)$

| $s_k$ | $s_0$ | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|---|
| $p_k$ | 0.4 | 0.3 | 0.2 | 0.1 |
| $I(s_k)$ bits | 1.322 | 1.737 | 2.322 | 3.322 |

5.
(a) Huffman Tree:




| Symbol | Code | Length |
|---|---|---|
| $s_0$ | 0 | 1 |
| $s_1$ | 01 | 2 |
| $s_2$ | 11 | 2 |

(b). Average cw length $= 0.7(1) + 0.15(2) + 0.15(2)$

$= 0.7 + 0.3 + 0.3 = 1.3$ bits/symbol

Let the alphabetical sequence be : AABABBBABAABABBBABBABB

First step: Take the single letter A. Since we have not seen A, we take A first.

Sequence : A | AB | ABB | B | ABA | ABAB | BB | ABBA | BB |

Step 2: Take AB next

Step 3: Take ABB next

Step 4: Next take B

Step 5: Take ABA next

Step 6: Take ABAB next

Step 7: Take BB next

Step 8: Take ABBA next

Step 9: Take BB next

The last phrase in step 9 is a repeated one as seen already in Step 7. That's fine.

A → 0
B → 1

| Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Numerical representation: | ∅A | 1B | 2B | ∅B | 2A | 5B | 4B | 3A | 7 |
| Code sequence: | ,∅ | 1,1 | 10,1 | 00,1 | 010,0 | 101,1 | 100,1 | 011,0 | 0111 |