University Of Thessaly

Mobile and Pervasive Computing

ECE515

Problem Set 1

Authors:
Lefkopoulou Eleni-Maria - 2557
Christodoulos Pappas - 2605

December 11, 2020

Exercise 1

To begin with, observing the given plot someone can easily see that the Min_SAUD, a cache replacement policy claimed to be optimal considering many system parameters such as Invalidation policies, skewed access and etc, seems to underperform, in comparison to other cache replacement policies such as PIX. To understand why such a thing happens, we need to basically understand how Min_SAUD actually works. In comparison to PIX, which is a theoretical policy, and so the access probabilities and broadcast frequencies are well defined and known from the beginning of the experiment, Min_SAUD does not have such luxury. To be more specific, considering the paper introducing Min_SAUD, access probability, update rate, and broadcast frequencies are constantly updated and learned through users access and system patterns. Moreover the function used to learn those variables is the same as the function used in TCP protocol. Thus it takes some time, in order for the policy to adapt into the system and users patterns. In addition to that, Min_SAUD is a more complex replacement policy, it has to keep in the MU's memory a considerable amount of data and thus the more data an algorithm has to manage and keep up to date the larger the overhead penalty it gets.

Exercise 2

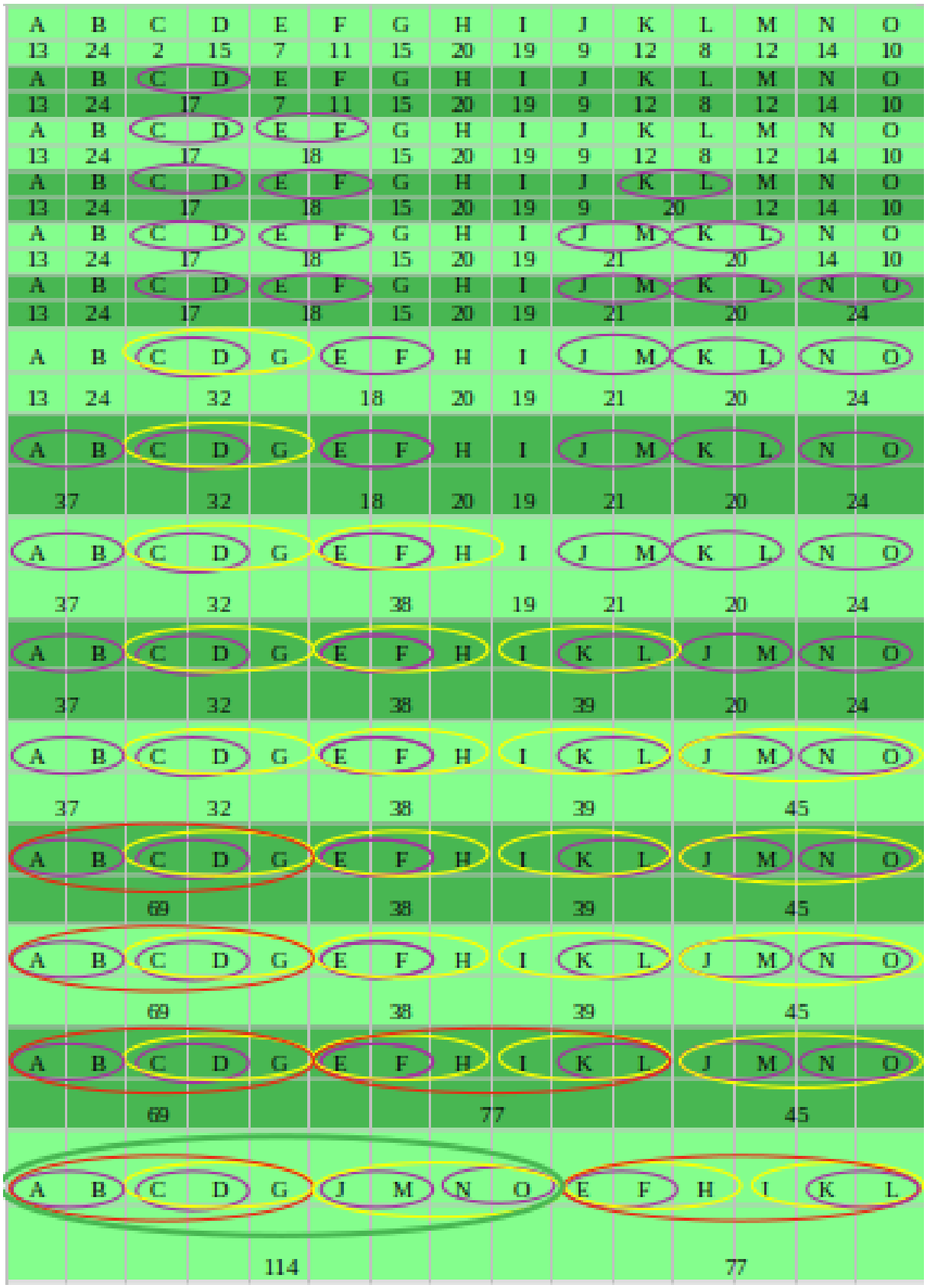
Without loss of generality, let us consider two items i and j, and their request rates D_i and D_j. Moreover, let T_i be the time needed to reply to a the request of the item i and T_j for the item j. Finally from the theory, in order for an sheduling algorithm to fulfill the Square-Root theorem, there must be true that $G(i) = G(j)$. So we have:

- For the RxW, we have: the number of requests of the item i in time T_i is $T_i \cdot D_i$, so $G(i) = T_i \cdot T_i \cdot D_i = T_i^2 \cdot D_i$, thus there must $G(i) = G(j)$ or $T_i^2 \cdot D_i = T_j^2 \cdot D_j$ or $\frac{T_i^2}{T_j^2} = \frac{D_j}{D_i}$ or $\frac{T_i}{T_j} = \sqrt{\frac{D_j}{D_i}}$ which is true according to the square root theorem.
- For the LWF, $G(i) = \sum_{k=1}^{T_i} k \cdot D_i = D_i \sum_{k=1}^{T_i} k = D_i \frac{T_i(T_i+1)}{2}$, thus there must $G(i) = G(j)$ or $\frac{T_i(T_i+1)}{T_j(T_j+1)} = \frac{D_j}{D_i} \neq \frac{T_i^2}{T_j^2}$.

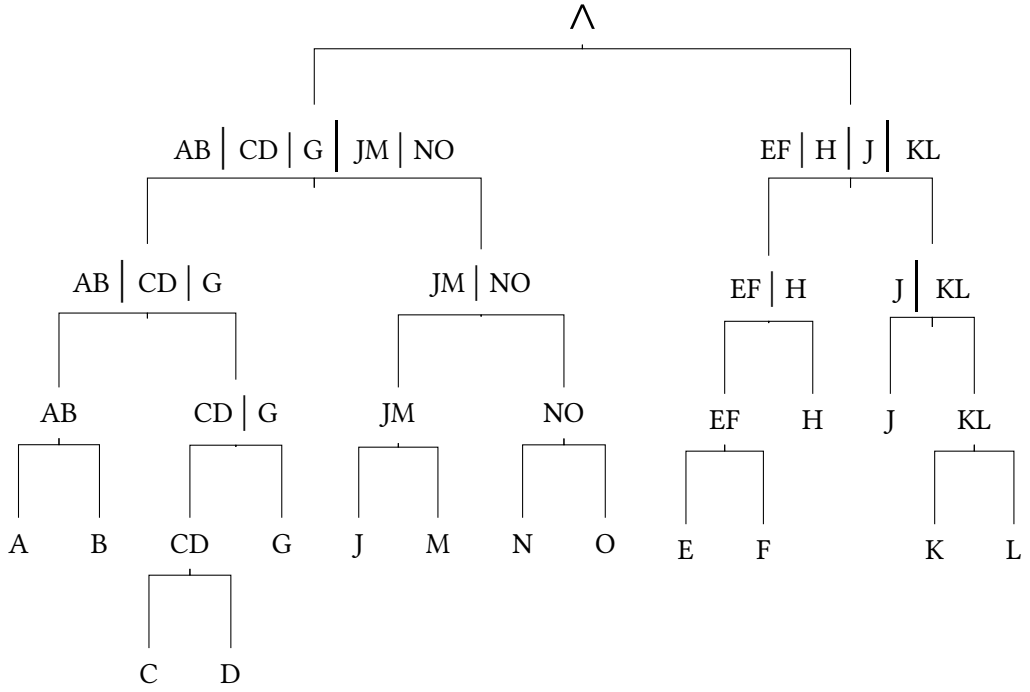
To conclude the LWF algorithm diverges from the Square-Root theorem.

Exercise 3

Minimun compatible pairs



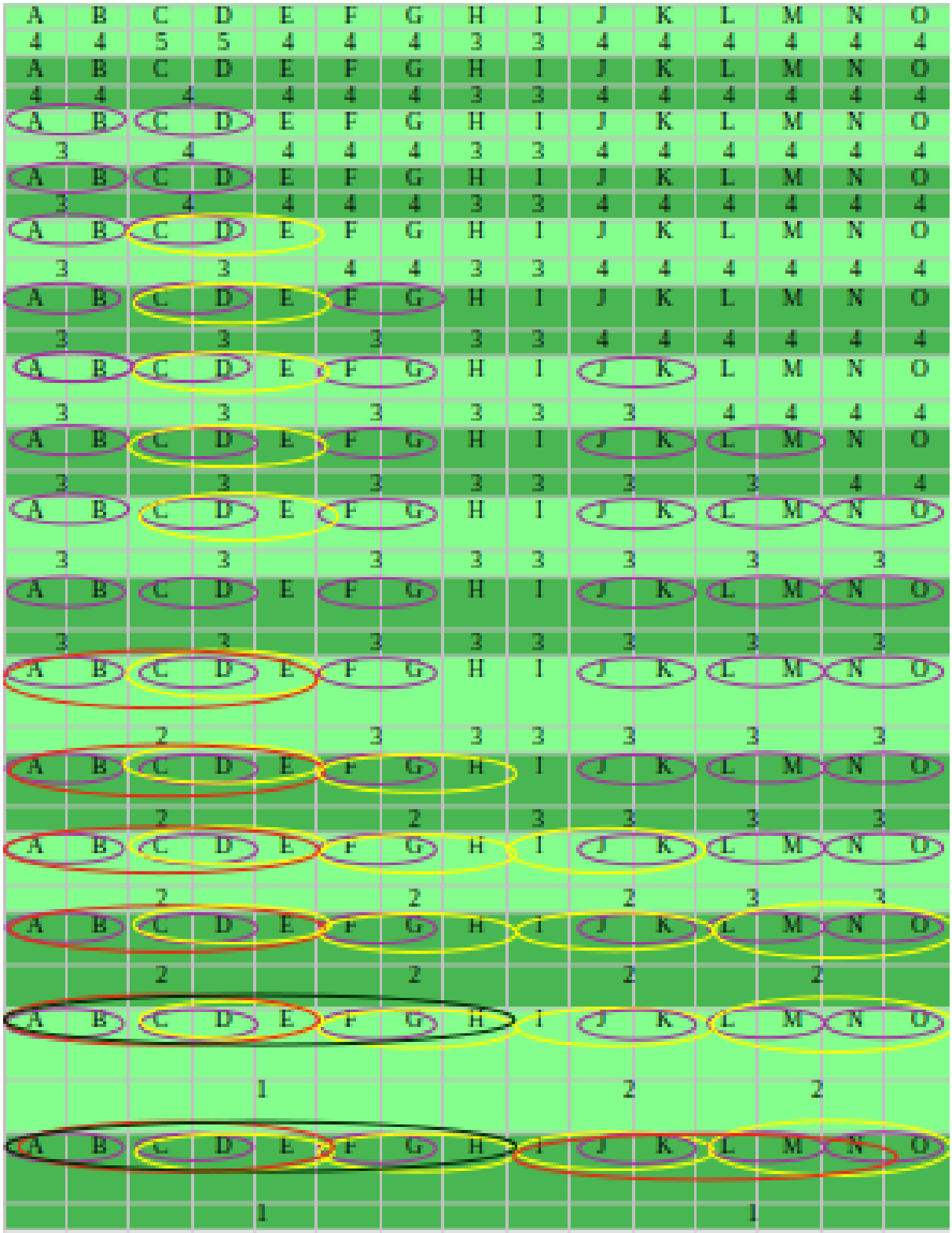
We design a tree based on the minimun compatible pairs to found the level of its letter.



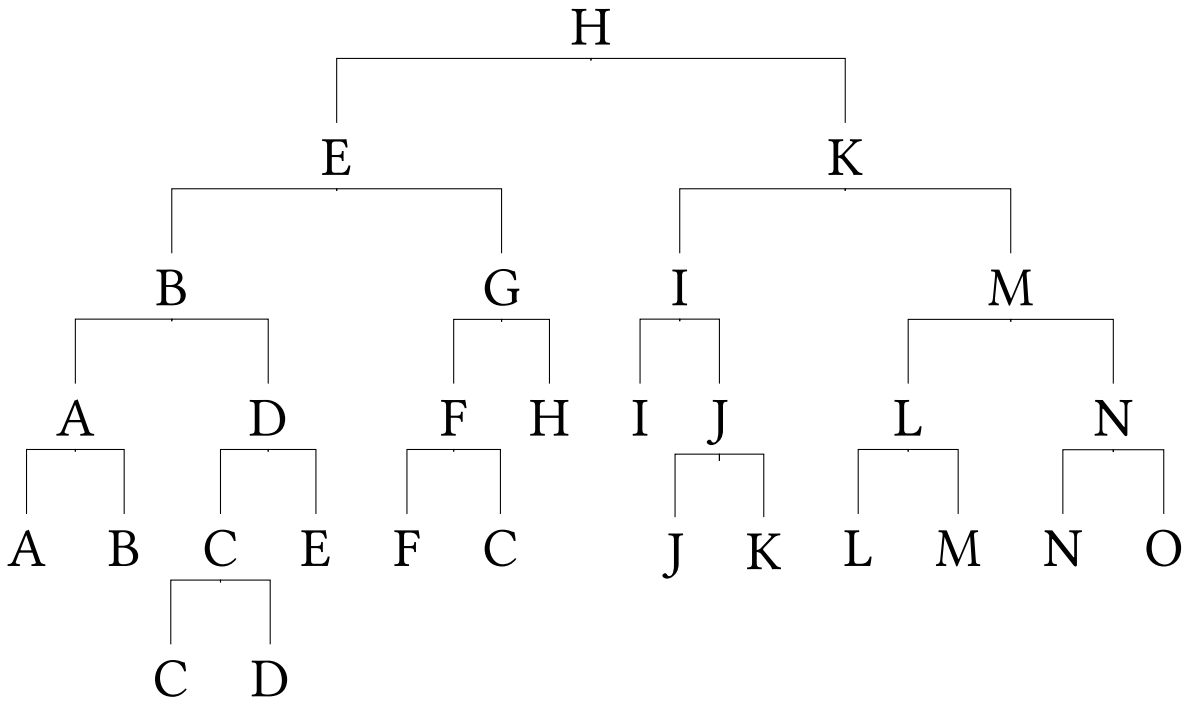
So the levels of the letters are the following:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4	4	5	5	4	4	4	3	3	4	4	4	4	4	4

Now have to make mergers according to the levels



The index will be in the following format:



The broadcast will be as follows (in the pointers after the letters the upper is for the youngest or equal child and the lower for the older)

H	$\frac{1}{16}$	E	$\frac{1}{10}$	C	$\frac{1}{6}$	A	$\frac{1}{2}$	A	B	$\frac{1}{2}$
B	C	D	$\frac{1}{2}$	D	E	F	$\frac{1}{2}$	F	G	$\frac{1}{2}$
G	H	K	$\frac{1}{6}$	I	$\frac{1}{2}$	I	J	$\frac{1}{2}$	J	K
N	$\frac{2}{1}$	O	M	$\frac{2}{1}$	N	L	$\frac{1}{2}$	L	M	

Exercise 4

As we consider,there can be at least two probable ways such that the server can organize the BitSequences-based IR, so that users can selectively tune into the broadcasting of the IR in order to get the information they need. A straightforward way is to give knowledge about the time or to be more specific the offset of each bit sequence. This can be achieved by having the server provide the offset of the sequence at the beginning of the Invalidation Report, side by side to the bit sequence timestamps. As a result the mobile user can select the appropriate bit-

sequence and then stay in dose mode until she has to tune into and retrieve the the report. Another way the server can organize the bitsequences-based IR so that it can be energy efficient is by doing the following :

- Reorder the IR such that the sequences will be broadcasted with the following order: $B_n B_{n-1}.., B_0$
- Let us propose a new sequence B'_i derived from B_i , which will be explained bellow, and let the server broadcasts $B'_n B'_{n-1}.., B'_0$ instead of $B_n B_{n-1}.., B_0$
- For each bit in the sequence, append a number of size $\frac{log(|B_i|)}{2}$, which gives us the information on how many 1's have been observed in the B_i until now, for examble for $B_i = 0101$ we will have a transformed sequence such that $B'_i = 00|10|01|11$.
- By knowing the above information the user can tune into the position of the data item in B'_{i-1} if it is needed.
- To give an illustration of how such search in invalidation report works, let an user is interested to find the validity of a data item d, until the timestamp of B'_k. Then the user first tunes in order to listen for the data item d. If it is 1 then she listens to the number of observed ones in order to find the d's position in B'_{n-1} and stays. The process stops when the user finds 0, or reaches in B'_k and the report in d's position is 1.

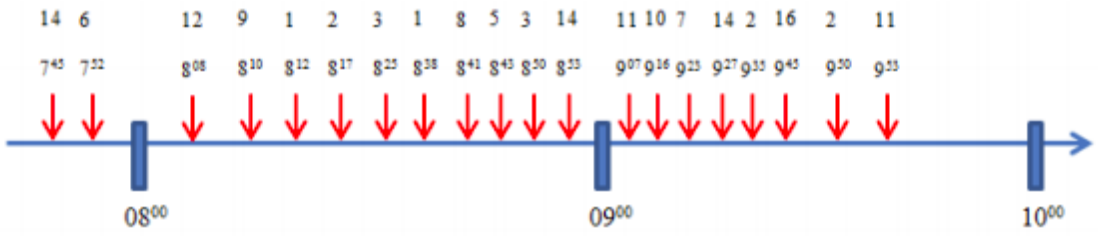
Exercise 5

current time = 10:00.

Client-1: Cache= {8, 14, 7, 2} Pending_reqs{5, 14} Tlb= 08:30

Client-2: Cache= {5, 9, 12, 1} Pending_reqs{5, 9, 15} Tlb= 08:50

Client-3: Cache= {2, 7, 10, 6} Pending_reqs{2, 4, 7, 10} Tlb=09:30



At the current time, the Invalidation Report will be broadcast in the BitSequences form. We must constract the IR in bitsequence format that the server should broadcast.

BS4 : 8:43

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	1

BS3 : 9:27

1	0	0	0	0	1	1	1

BS2 : 9:50

1	1	0	0

BS1 : 9:53

0	1

BS0 : 10:00

0

IR$_{BS}$: 10:00 9:53 9:50 9:27 8:43 0 01 1100 10000111 0110101001100101

Based on the BitSequences logic that we have already built into the above steps we can examine which cache data each client can use and which he can drop out of cache.

Client-1 :Tlb = 8:30 => Tlb < TS(B4)

So the client must drop all the elements from cache.

Client-2 :Tlb = 8:50 => TS(B4) < Tlb < TS(B3)

So the client must drop the 5 from cache and he will keep 9 and 15, because these elements have not any change at this timestamp.

Client-3 :Tlb = 9:30 => TS(B3) < Tlb < TS(B2)

So the client must drop the 2 from cache and he will keep 7,10 and 6, because these elements have not any change at this timestamp.

Exercise 6

Given the gain formula of the Min_SAUD , $G(i) = \frac{p_i}{s_i}(\frac{b_i}{1+x_i} - u)$, we make the following observations:

- If the items are read-only then for each u_i it follows that $u_i = 0$ and u = 0. So the gain becomes $G(i) = \frac{p_i \cdot b_i}{s_i}$.
- intuitively we can see that $b_i = 1/f_i$. So the gain transforms onto $G(i) = \frac{p_i}{s_i \cdot f_i}$.
- Finally, when the size of the files is the same, for reasons of simplicity we set $s_i = 1$, then we have $G(i) = \frac{p_i}{f_i}$.

Thus, we can see that Min_SAUD transforms to PIX if the data sizes are the same and also they are read-only.

Exercise 7

1ST APPROACH

The server has the below elements with their popularity

a: 0.40 b: 0.25 c: 0.15 d: 0.10 e: 0.10

The last broadcast program was the following : abedaabbcd. Examining the popularity of each element we decide to create a new broadcast programm using 3 broadcast disks.

Δ_1= {a}

Δ_2= {b}

Δ_3= {c,d,e}

We choose the following frequencies for each disk f_1=4 f_2=2 f_3=1. Next we calculate the least common multiple(LCM) of two frequencies.

LCM = {1,2,4} = 4

So now we can calculate the chunks for each broadcast disk.

#chunks = LCM ÷ frequency
$chunk_{\Delta 1} = \frac{4}{4} = 1$ {a}
$chunk_{\Delta 2} = \frac{4}{2} = 2$ {b},{_}
$chunk_{\Delta 3} = \frac{4}{1} = 4$ {d},{e},{c},{_}
So 1 cycle of the broadcast program will be : abca_dabea_ _ and this will be repeated.The client start listen at the time '0' and his cache with size 2 has already 2 elements inside : c and d.
Not normalized popularity of each element on the client is
a: 5 b: 4 c: 3 d: 2 e: 1

The frequency of its element are :

- $f_a = \frac{4}{12} = \frac{1}{3}$
- $f_b = \frac{2}{12} = \frac{1}{6}$
- $f_c = \frac{1}{12}$
- $f_d = \frac{1}{12}$
- $f_e = \frac{1}{12}$

The policy that cache follows is PIX . PIX values for each element are the following :

- $PIX_a = 5 \div \frac{1}{3} = 15$
- $PIX_b = 4 \div \frac{1}{6} = 24$
- $PIX_c = 3 \div \frac{1}{12} = 36$
- $PIX_d = 2 \div \frac{1}{12} = 24$
- $PIX_e = 1 \div \frac{1}{12} = 12$

The elements that are requested ,the cache hit or miss, the latency,the cache changes and the current broadcasting element are shown in the table bellow.

element	hit	latency	cache	program
a	✗	1	cd	**a**bca_dabea_ _
b	✗	1	cb	a**b**ca_dabea_ _
e	✗	7	cb	abca_dab**e**a_ _
d	✗	9	cd	abca_**d**abea_ _
a	✗	1	cd	abca_dabea_ _
a	✗	3	cd	abca_dabe**a**_ _
b	✗	4	cb	a**b**ca_dabea_ _
b	✓	0	cb	abca_dabea_ _
c	✓	0	cb	abca_dabea_ _
d	✗	4	cd	abca_**d**abea_ _

To conclude the hit ratio of the cache is $\frac{2}{10}$ = 0,2 = 20%.
The total latency is total_lat = 1+1+7+9+1+3+4+0+0+4 =30
so the average latency is avg_latency = total_lat÷10 = 3.

We can also use the square root theorem to calculate frequencies for each disk fi $\propto\sqrt{\frac{pi}{li}}$ .With this method the broadcast programm will be a b d _ c e a _ _ _ b _ a c _ _ _ _ and the frequencies of the disks will be f_1=3 f_2=2 f_3=1 . With this way the average latency will be 3,9 and 40% hit rate.So the way that we choose gave us a better latency

2ND APPROACH: different grouping of elements in broadcast disks

Δ_1= {a}

Δ_2= {b,c}

Δ_3= {d,e}

We choose the following frequencies for each disk f_1=4 f_2=2 f_3=1. Next we calculate the least common multiple(LCM) of two frequencies.

LCM = {1,2,4} = 4

So now we can calculate the chunks for each broadcast disk.

#chunks = LCM ÷ frequency

$chunk_{\Delta 1} = \frac{4}{4} = 1$ {a}

$chunk_{\Delta 2} = \frac{4}{2} = 2$ {b},{c}

$chunk_{\Delta 3} = \frac{4}{1} = 4$ {d},{e},{_},{_}

So 1 cycle of the broadcast program will be a b d a c e a b _ a c _ and this will be repeated.The client start listen at the time '0' and his cache with size 2 has already 2 elements inside : c and d.

Not normalized popularity of each element on the client is

a: 5 b: 4 c: 3 d: 2 e: 1

The frequency of its element are : $f_a = \frac{4}{12} = \frac{1}{3}$

$f_b = \frac{2}{12} = \frac{1}{6}$

$f_c = \frac{2}{12} = \frac{1}{6}$

$f_d = \frac{1}{12}$

$f_e = \frac{1}{12}$

The policy that cache follows is PIX . PIX values for each element are the following :

$PIX_a = 5 \div \frac{1}{3} = 15$

$PIX_b = 4 \div \frac{1}{6} = 24$

$PIX_c = 3 \div \frac{1}{6} = 18$

$PIX_d = 2 \div \frac{1}{12} = 24$

$PIX_e = 1 \div \frac{1}{12} = 12$

element	hit	latency	cache	program
a	✗	1	cd	**a**bdaceab_ac_
b	✗	1	bd	a **b**daceab_ac_
e	✗	4	bd	abdac**e**ab_ac_
d	✓	0	bd	abdac**e**ab_ac_
a	✗	1	bd	abdace**a**b_ac_
a	✗	3	bd	abdaceab_**a**c_
b	✓	0	bd	abdaceab_**a**c_
b	✓	0	bd	abdaceab_**a**c_
c	✗	1	bd	abdaceab_a**c**_
d	✓	0	bd	abdaceab_a**c**_

To conclude the hit ratio of the cache is $\frac{4}{10} = 0{,}4 =$ 40%.
The total latency is total_lat = 1+1+4+0+1+3+0+0+1+0 =11
so the average latency is avg_latency = total_lat ÷10 = 1,1 .

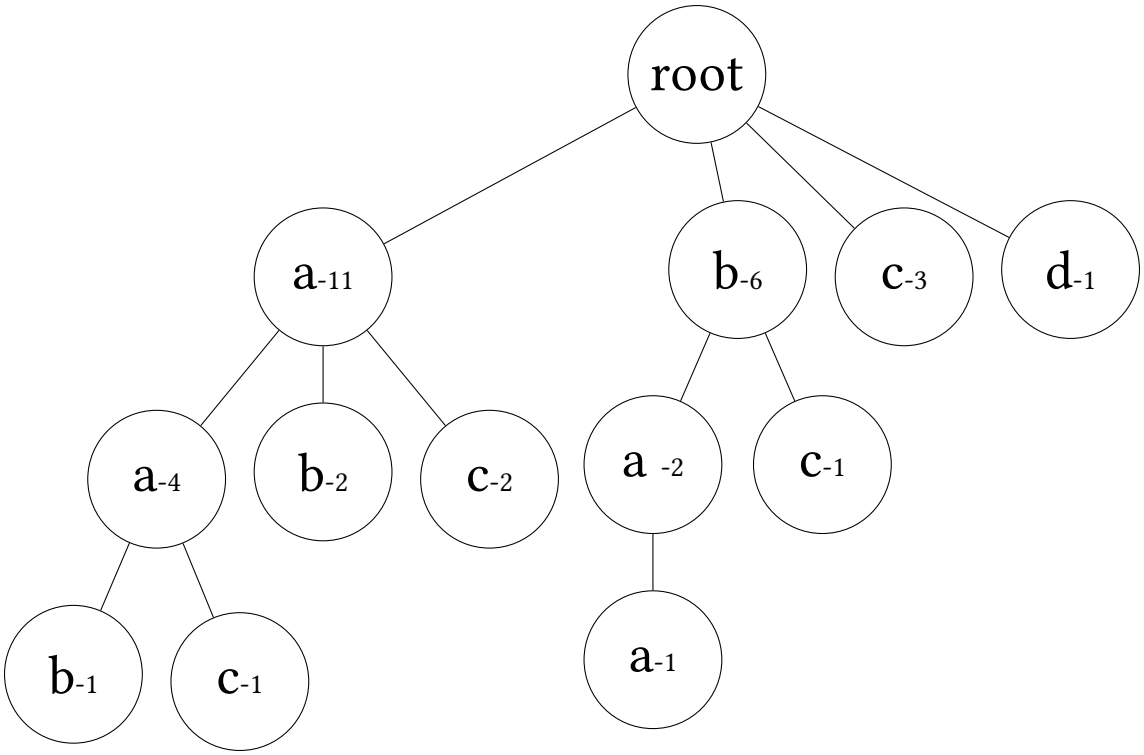
Exercise 8

I found with LZ parsing phrases and suffixes to create the augmented tree.

We have the sequence aabbcaaaacdaaddadcdaa

So the phrases and the suffixes are :

a		
ab,	b	
b		
aa,	a	
aac,	ac,	c
d		
aab,	ab,	b
ba,	a	
bc	c	
ba	a	
bc	c	
baa,	aa,	a



We want to forecast what will be broadcast after baa so we will use the probabilities of phrases from context-2 context-1 and context-0

The phrases and the probability counters are the following (∧-> deficit)

aa (order-2)	a(order-1)	∧(order-0)
b|aa(1)	a|a(2)	a(4) aac(1)
c|aa(1)	b|a(2)	b(3) aab(2)
∧ |aa(2)	c|a(1)	c(3) ac(1)
	ab|a(1)	d(1) ba(1)
	ac|a(1)	aa(2) bc(1)
	∧|a(4)	aab(1) baa(1)

Calculation of the probabilities of occurrence

Posibilities	a	b	c	d
b : $\frac{1}{4}+\frac{2}{4}[\frac{2}{11}+\frac{4}{11}*\frac{3}{21}]$ = 0,367	-	0,367	-	-
c : $\frac{1}{4}+\frac{2}{4}[\frac{1}{11}+\frac{4}{11}*\frac{3}{21}]$ = 0,32	-	-	0,32	-
a : $\frac{2}{4}[\frac{2}{11}+\frac{4}{11}*\frac{4}{21}]$ = 0,125	0,125	-	-	-
ab : $\frac{2}{4}[\frac{1}{11}+\frac{4}{11}*\frac{2}{21}]$ = 0,063	0,0315	0,0315	-	-
ac : $\frac{2}{4}[\frac{1}{11}+\frac{4}{11}*\frac{1}{21}]$ = 0,054	0,027	-	0,027	-
d : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	-	-	-	0,0086
aa : $\frac{2}{4}*\frac{4}{11}*\frac{2}{21}$ = 0,017	0,017	-	-	-
aab : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	0,0057	0,0028	-	-
aac : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	0,0057	-	0,0028	-
ba : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	0,0043	0,0043	-	-
bc : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	-	0,0043	0,0043	-
baa : $\frac{2}{4}*\frac{4}{11}*\frac{1}{21}$ = 0,0086	0,0057	0,008	-	-
SUM	0,3176	0,4379	0,3541	0,0086

To conclude the paging should be done with the following sequence b , c , a , d .

Exercise 9

To begin with, we observe that if there is no possibility that two items are going to be requested simultaneously, this can happen for example due to hardware limitations or consistency issues, then $P_{ij} = 0$ for every item i and j. Also P_{ii} can be written simply as P_i. Thus the cost function becomes $G_i = \tau_i \cdot P_i$, where is the same as the PT's prefetching heuristic.
