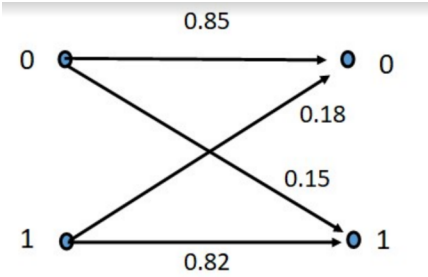
UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE CIENCIAS
TEORIA DE CODIFICACIÓN

Edgar Santiago Ochoa Quiroga
María Alejandra Rodríguez Ríos

Ejercicio 1
Suponga que una fuente genera dígitos binarios con distribución uniforme, los mensajes son enviados a través de un canal cambia los símbolos que genera una fuente binaria con las probalidades que se muestran en la gráfica.



Responda cada una de las siguientes preguntas justificando su razonamiento:
a). ¿Cuál es el número más probable de errores que se puede encontrar en un mensaje de 5000 bits que pase por el canal?

Solución. Teniendo en cuenta que al evaluar si un bit fue bien codificado através del canal, el resultado que ponemos obtener es sí o no, podemos resolver este ejercicio utilizando la distribución binomial.
Sabemos que, los bits pueden tomar valores de 0 o 1, sin embargo, el enunciado nos dice que la manera en la que se distribuyen através de los 5000 bits es uniforme, es decir, que podemos tomar la probabilidad de que el bit sea 0 o 1 como $\frac{1}{2}$.
Ahora bien, para calcular cada uno de los errores tomamos en cuenta la información del gráfico, así obtenemos que

$$P(0|1) = 0{,}18$$
$$P(1|0) = 0{,}15$$

Por lo cual, el error total al codificar es

$$P(error) = \frac{1}{2} \cdot 0{,}18 + \frac{1}{2} \cdot 0{,}5 = 0{,}165$$

Luego, al calcular la esperanza tenemos que la cantidad de errores es

$$\mu = 5000 \cdot 0{,}165 = 825$$

Con lo cual podemos decir que el número más probable de errores es 825.

b). Si se codifica cada bit que genera la fuente triplicandolo, en qué porcentaje se reduce el número de errores? Suponga que para la decodificación se usa mayoría de bits por tripla.

Solución. Ahora, tenemos que calcular el error para una fuente que triplica los bits, sabemos que mediante esta codificación para que se de un error quiere decir que 2 o 3 bits al pasar por el canal se codificaron de manera incorrecta. Así que calculemos esa probabilidad

$$P(errortripla) = P(2errores) + P(3errores)$$

Al igual que en el numeral anterior, vamos a utilizar la distribución binomial para calcular la probabilidad de que cambien 2 o 3 bits

$$P(2 \text{ errores}) = \binom{3}{2} p^2(1-p) = 3 \cdot (0{,}165)^2(0{,}835) = 0{,}0681986$$
$$P(3 \text{ errores}) = \binom{3}{3} p^2(1-p) = (0{,}165)^3 = 0{,}004492.$$

Luego el error por tripla tenemos que es

$$\begin{aligned} P(errortripla) &= P(2errores) + P(3errores) \\ &= 0{,}0681986 + 0{,}004492 \\ &= 0{,}0726906. \end{aligned}$$

Por lo cual tenemos que la reducción porcentual del error es

$$Reducciónporcentual = \frac{0{,}165 - 0{,}0726906}{0{,}165} \cdot 100 = 55{,}9451\,\%.$$

c). Si se codifica cada bit que genera la fuente con una n-tupla y se usa mayoría de bit por n tupla en la decodificación, cuál sería la longitud mínima n - tupla para obtener por lo menos un 95 % de certeza en la decondificación?

Solución. Para realizar este ejercicio tenemos que tener en cuenta que n para que el criterio se decidible debe ser impar y por el numeral anterior tenemos cuando $n = 3$ el error es de 7,269 %, como estamos buscando que el error sea de 5 % o menor vamos a probar los siguientes casos

- **Caso** $n = 5$ Para este caso vamos a aplicar la distribución de binomial ahora tomando los

casos donde se presenten por lo menos 3 errores, asi calculamos lo siguiente

$$P(3\text{ errores}) = \binom{5}{3} p^3(1-p)^{5-3} = 10(0{,}165)^3(0{,}835)^{5-3} = 0{,}0313202,$$
$$P(4\text{ errores}) = \binom{5}{4} p^4(1-p)^{5-4} = 5(0{,}165)^4(0{,}835)^{5-4} = 0{,}00309451,$$
$$P(5\text{ errores}) = \binom{5}{5} p^5(1-p)^{5-5} = (0{,}165)^5(0{,}835) = 0{,}000122298$$

Entonces,tenemos que el error del código que generan las 5−tuplas es

$$\begin{aligned} P(5\texttt{tuplaserror}) &= P(3\text{ errores}) + P(4\text{ errores}) + P(5\text{ errores}) \\ &= 0{,}0313202 + 0{,}00309451 + 0{,}000122298 \\ &= 0{,}034537. \end{aligned}$$

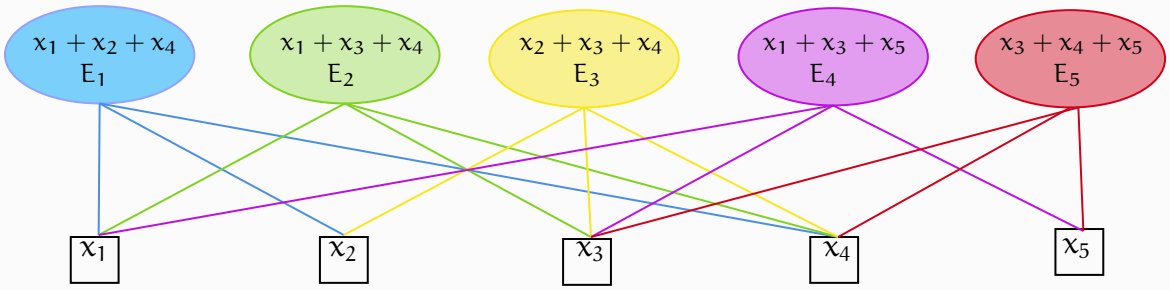
Luego el error es de 3,4537 % y por lo tanto la longitud mínima para que la certeza sea de 95 % es de 5.

Ejercicio 2
Se codifican los bits de una fuente de acuerdo a la siguiente función

$$\texttt{Cod}(x_1, x_2, x_3, x_4, x_5) = (x_1, x_2, x_3, x_4, x_5, x_1{+}x_2{+}x_4, x_1{+}x_3{+}x_4, x_2{+}x_3{+}x_4, x_1{+}x_3{+}x_5, x_3{+}x_4{+}x_5)$$

Si el mensaje recibido es 1101110101 determine si hay un error en el mensaje y si es posible corregir el error . Use diagramas de Ven o el método de Galager para llegar a la solución. Deje todos los cálculos que le llevan a la solución en su hoja de respuesta.

Solución. Para resolver este ejercicio vamos a utilizar el método de Galager, por lo cual, de acuerdo con las ecuaciones que nos da la codificación tenemos el siguiente gráfico



Luego tenemos que para ver que bits cumplen las ecuaciones y ver si debemos cambiar alguno realizamos el siguiente cálculo

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	1	–	1	–	0
$x_2 = 1$	1	–	0	–	–	1
$x_3 = 0$	–	1	0	1	0	2
$x_4 = 1$	1	1	0	–	0	2
$x_5 = 1$	–	–	–	1	0	1

Teniendo en cuenta esos valores, vamos a empezar a realizar los cambios en los valores de la primera columna, puesto que suponemos que las sumas tienen el bit correcto, para facilidad de lectura cada tabla de operaciones tendra el nombre de un paso para asi poder devolvernos de manera sencilla en caso de que se presenten bucles.
Paso 1: Cambiamos el valor de x_3, de acuerdo con el criterio podemos elegir cualquiera entre x_3 y x_4, pero en este caso elegimos x_3

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	0	–	0	–	2
$x_2 = 1$	1	–	1	–	–	0
$x_3 = 1$	–	0	1	0	1	2
$x_4 = 1$	1	0	1	–	1	1
$x_5 = 1$	–	–	–	0	1	1

Como el criterio dice que debemos cambiar alguno de los que más presenta errores y cambiar x_3 es devolverse al problema inicial, entonces cambiemos x_1.
Paso 2: Cambiamos el valor de x_1 y como resultado tenemos que

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	0	1	–	1	–	1
$x_2 = 1$	0	–	1	–	–	1
$x_3 = 1$	–	1	1	1	1	0
$x_4 = 1$	0	1	1	–	1	1
$x_5 = 1$	–	–	–	1	1	0

En este caso no podemos volver a cambiar el valor de x_1 porque estaríamos devolviendonos un paso, por lo cual vamos a elegir cambiar x_2, se podría hacer el cambio con x_4 sin embargo lo haremos en caso de ser necesario más adelante.
Paso 3: Cambiamos el valor de x_2 y realizamos los cálculos obteniendo que

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	1	–	1	–	0
$x_2 = 0$	1	–	0	–	–	1
$x_3 = 1$	–	1	0	1	1	1
$x_4 = 1$	1	1	0	–	1	1
$x_5 = 1$	–	–	–	1	1	0

En este caso no podemos cambiar x_2 porque tendiamos un paso ya estudiado (Paso 2) y por los criterios aunque podemos elegir a x_3 o x_4 vamos a elegir a x_4.
Paso 4: Cambiamos el valor de x_4 y calculamos

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	0	0	–	1	–	2
$x_2 = 0$	0	–	1	–	–	1
$x_3 = 1$	–	0	1	1	0	2
$x_4 = 0$	0	0	1	–	0	3
$x_5 = 1$	–	–	–	1	0	1

Aqui el algoritmo nos dice que tenemos que cambiar el valor de x_4 y eso sería devolverse al paso inmediatamente anterior, por lo cual tenemos un **bluce** para este caso. Por lo cual retomamos las opciones que nos daba el (Paso 3), es decir cambiar el valor de x_3

Paso 5: Volvemos un paso y cambiamos x_3

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	0	–	0	–	2
$x_2 = 0$	1	–	1	–	–	0
$x_3 = 0$	–	0	1	0	0	3
$x_4 = 1$	1	0	1	–	0	2
$x_5 = 1$	–	–	–	0	0	2

Aqui el algoritmo nos dice que tenemos que cambiar el valor de x_3 y eso seria devolverse al (Paso 2) por lo cual entramos en un **bucle**, al finalizar el (Paso 2) elegimos cambiar el valor de x_2 y eso nos llevó a un **bucle** por lo cual vamos a devolvernos y cambiar esa escogencia y vamos a cambiar x_4

Paso 6: Volvemos al final del paso 2 y cambiamos x_4

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	0	–	1	–	1
$x_2 = 1$	1	–	0	–	–	1
$x_3 = 1$	–	0	0	1	0	3
$x_4 = 0$	1	0	0	–	0	3
$x_5 = 1$	–	–	–	0	0	2

En este caso solo podemos cambiar el valor de x_3, porque como lo hemos explicado anteriormente cambiar x_4 seria devolvernos un paso.

Paso 7: Cambiamos el valor de x_3

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	1	–	0	–	1
$x_2 = 1$	1	–	1	–	–	0
$x_3 = 0$	–	1	1	0	1	1
$x_4 = 0$	1	1	1	–	1	0
$x_5 = 1$	–	–	–	0	1	1

En este caso podemos hacer el cambio para el siguiente paso con x_1 y x_5 para este caso escogemos x_1

Paso 8: Cambiamos el valor de x_1

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	0	0	–	1	–	2
$x_2 = 1$	0	–	1	–	–	1
$x_3 = 0$	–	0	1	1	1	1
$x_4 = 0$	0	0	1	–	1	2
$x_5 = 1$	–	–	–	1	1	0

Paso 9: Cambiamos el valor de x_4

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	1	–	1	–	0
$x_2 = 1$	1	–	0	–	–	1
$x_3 = 0$	–	1	0	1	0	2
$x_4 = 1$	1	1	0	–	0	2
$x_5 = 1$	–	–	–	1	0	1

Si realizo el cambio por x_3 volvería al (Paso 1) y entraría en un **bucle** y si cambio por x_4 también entraría en **bucle** puesto que volveria al (Paso 8). Por lo cual me devuelvo al final del (Paso 7) y realizo el cambio a x_5 en lugar de a x_1

Paso 10: Vuelvo al final del (Paso 7) y realizo el cambio al valor de x_5

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	1	–	1	–	0
$x_2 = 1$	1	–	1	–	–	0
$x_3 = 0$	–	1	1	1	0	1
$x_4 = 0$	1	1	1	–	0	1
$x_5 = 0$	–	–	–	1	0	1

Por un argumento similar a los anteriores sabemos que no podemos hacer el cambio del valor a x_5 y escogemos cambiar el valor de x_3
Paso 11: Cambio al valor de x_3

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	1	0	–	0	–	1
$x_2 = 1$	1	–	0	–	–	1
$x_3 = 1$	–	0	0	0	1	3
$x_4 = 0$	1	0	0	–	1	2
$x_5 = 0$	–	–	–	0	1	1

Aqui el algoritmo nos dice que cambiemos el valor de x_3 pero eso seria devolvenos al paso anterior, por lo cual estamos en un **bucle**. Por lo cual nos devolvemos al final del (Paso 10) y escogemos cambiar el valor de x_4
Paso 12: Vuelvo al final del (Paso 10) y realizo el cambio al valor de x_4

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 0$	0	0	–	1	–	2
$x_2 = 1$	0	–	0	–	–	2
$x_3 = 0$	–	0	0	1	1	2
$x_4 = 1$	0	0	0	–	1	3
$x_5 = 0$	–	–	–	1	1	0

Al igual que en el el (Paso 11) entro en un **bucle** puesto que el algoritmo me da como resultado volver a cambiar el valor de x_4. Por lo cual debo devolverme en pasos hasta donde hice una escogencia de la cual no he probado posibilidades y eso me lleva a la tabla inicial (Paso 0) y ahi eligiriamos cambiar el valor de x_4 en lugar a x_3 sin embargo, esto presenta el mismo caso evidenciado en el (Paso 8) por lo cual entriamos en **bucle** y por lo cual, bajo el criterio de puntución

teniendo en cuenta el que más falla no podriamos resolverlo y solo seria un algoritmo que notifica el error, ahora, mirarmos ahora el caso con los que presentan solo un error y un acierto, probemos con x_2

Paso 13: Probamos cambiando a x_2

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	0	1	–	1	–	1
$x_2 = 0$	0	–	1	–	–	1
$x_3 = 0$	–	1	1	1	0	1
$x_4 = 1$	0	1	1	–	0	2
$x_5 = 1$	–	–	–	1	0	1

Como tenemos que el x_4 es el que tiene mayor cantidad de errores vamos a cambiarlo.

Paso 14: Probamos cambiando a x_4

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	0	–	1	–	1
$x_2 = 0$	1	–	0	–	–	1
$x_3 = 0$	–	0	0	1	1	2
$x_4 = 0$	1	0	0	–	1	2
$x_5 = 1$	–	–	–	1	1	0

Por razones análogas ya explicadas anteriormente solo podemos elegir para cambiar a x_3

Paso 15: Probamos cambiando a x_3

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	1	–	0	–	0
$x_2 = 0$	1	–	1	–	–	0
$x_3 = 1$	–	1	1	0	0	2
$x_4 = 0$	1	1	1	–	0	1
$x_5 = 1$	–	–	–	0	0	2

Como x_5 es el que tiene más errores y no se cambió en el paso inmediatamente anterior entonces el criterio nos dice que lo cambiemos

Paso 16: Probamos cambiando a x_5

x	$E_1 = 1$	$E_2 = 0$	$E_3 = 1$	$E_4 = 0$	$E_5 = 1$	Ceros totales
$x_1 = 1$	1	1	–	1	–	0
$x_2 = 0$	1	–	1	–	–	0
$x_3 = 1$	–	1	1	1	1	0
$x_4 = 0$	1	1	1	–	1	0
$x_5 = 0$	–	–	–	1	1	0

Con lo que tenemos que el codigo original antes de cambiar por el canal era 10100 y por lo cual habia errores en x_2, x_3, x_4 y x_5. Y el mensaje corregido es 1010010101

Ejercicio 3

Clasifique los siguientes códigos de tal manera que puedan codificar la misma fuente. Luego indique por fuente cuál sería el mejor código. Justifique todas sus afirmaciones.

- $C_1 = \{0, 1\}$
- $C_2 = \{00, 01, 10\}$
- $C_3 = \{000, 111\}$
- $C_4 = \{000, 011, 101, 110, 001, 100\}$
- $C_5 = \{00000, 11111, 22222\}$
- $C_6 = \{001, 010, 012, 021, 100, 221\}$

Solución. Vamos a realizar la clasificación teniendo en cuenta la cantidad de símbolos del alfabeto, por lo cual realizaremos la comparación C_1 y C_3, C_2 y C_5 y C_4 y C_6.

- C_1 y C_3: Tenemos que ambos códigos son binarios y univocamente decodificables, la longitud del primero es menor que el del segundo, sin embargo, teniendo en cuenta que el canal puede tener ruido, el código C_1 no puede hacer la detección de errores a diferencia del C_3 y tomando como ejemplo el punto 1, vemos que la reducción del error en la codificación si bien depende de la probabilidad en el canal se evidencia para ese caso en 55 %. Por lo cual el mejor código para codificar esa fuente es C_3.
- C_2 y C_5: Ambos códigos son libres de prefijos y por ende univocamente decodificables, sin embargo, de manera análoga al item anterior tenemos que C_2 es un código que no detecta ni corrige errores, pero, C_5 es capaz de detectar códigos hasta con 4 errores y corregir códigos que tienen a lo más 2 errores. Por lo cual C_5 es un código más robusto y en este caso es mejor.
- C_4 y C_6: Tenemos que ninguno de los códigos funciona para detectar o corregir errores, pero el código C_4 al ser más completo y por la manera en la que está dado siempre va a tener una respuesta ya sea que el código este bien o mal codificado, sin embargo, al ser C_6 un código triario que es menor completo y la manera en la que está construido, existe la posibilidad de que al pasar por el canal no exista un elemento en el alfabeto con el cual emparejarlo, lo que permitiria que por lo menos en algunos se evidencie el error aunque no sea corregible, hace que sea mejor para pasar por un canal con ruido.

Ejercicio 4
Determine justificando, si los códigos $C_1 = \{1, 10, 100, 1000\}$ y $C_2 = \{1, 01, 001, 0001\}$ son univocamente decodificables.

Solución. Procederemos con cada código por separado

- Sea $C = \{1, 10, 100, 1000\}$., Donde nuestro alfabeto para codificar es $D = \{0, 1\}$ Procedamos por medio se Sardinas-Patterson. Primero definimos $C_0 = C$, por el algoritmo tenemos que

  $$C_1 = \{w \in D^+ | uw = v, \text{ con } u \in C_0 \text{ y } v \in C \text{ o } u \in C \text{ y } v \in C_0\}.$$

  Note que como $C_0 = C$ basta con resolver ecuaciones de concatenación de palabras donde solo revisamos los elementos de C. Procedemos por casos.

Caso 1: Si $u = 1000$, tenemos las siguientes cuatro ecuaciones

$$1000w = \begin{cases} 1 \\ 10 \\ 100 \\ 1000 \end{cases}$$

Note que $|1000w| \geq 5$ ya que $|w| \geq 1$. Pero las longitudes de las palabras al otro lado de la igualdad son a lo más 4, por lo que estas ecuaciones no tienen solución.

Caso 2: Si $u = 100$, tenemos las siguientes cuatro ecuaciones

$$100w = \begin{cases} 1 \\ 10 \\ 100 \\ 1000 \end{cases}$$

Para las primeras tres al igual que en el caso anterior no hay solución simplemente viendo las longitudes, pero para la ultima tomando $w = 0$, tenemos nuestra primera solución.

Caso 3: Si $u = 10$, tenemos las siguientes cuatro ecuaciones

$$10w = \begin{cases} 1 \\ 10 \\ 100 \\ 1000 \end{cases}$$

Siguiendo la lógica de los anteriores casos nos damos cuenta que $w = 00$ es solución de la cuarta ecuación, mientras que $w = 0$ de la tercera y las otras dos no tienen.

Caso 4: Si $u = 1$, tenemos las siguientes cuatro ecuaciones

$$1w = \begin{cases} 1 \\ 10 \\ 100 \\ 1000 \end{cases}$$

Siguiendo la lógica de los anteriores casos nos damos cuenta que $w = 000$ es solución de la cuarta ecuación, mientras que $w = 00$ de la tercera y $w = 0$ de la segunda, pero la primera no tiene solución.

Así tenemos que $C_1 = \{0, 00, 000\}$. Siguiendo el algoritmo tenemos que

$$C_2 = \{w \in D^+ | \, uw = v, \text{ con } u \in C_1 \text{ y } v \in C \text{ o } u \in C \text{ y } v \in C_1\}.$$

Veamos los posibles casos.

Caso 1: Si $u \in C_1$ y $v \in C$, se tiene que la palabra uw siempre empieza en 0, ya que u es igual a $0, 00$ o 000, mientras que todas las palabras v en C empiezan en 1, así ninguna ecuación tiene solución.

Caso 2: Si $u \in C$ y $v \in C_1$, pasa igual que en el caso anterior. Todas la posibles palabras uw empiezan por 1, ya que cualquier palabra en C empieza por uno, mientras que todas las palabras en C_1 empiezan por 0, así ninguna ecuación tiene solución. Así $C_2 = \varnothing$.

Por la construcción podemos concluir que $C_3 = C_4 = \cdots = \varnothing$. Por lo tanto

$$C_\infty = \bigcup_{i=1}^{\infty} C_i = C_1.$$

Así podemos notar que

$$C \cap C_\infty = \{1, 10, 100, 1000\} \cap \{0, 00, 000\} = \varnothing.$$

Concluyendo por el teorema de Sardinas-Patterson que C es unívocamente decodificable.

- Sea $C = \{1, 01, 001, 0001\}$, donde nuestro alfabeto para codificar es $D = \{0, 1\}$. Siguiendo el algoritmo, tomamos $C_0 = C$, siguiendo el algoritmo tenemos que

$$C_1 = \{w \in D^+ | \, uw = v, \text{ con } u \in C_0 \text{ y } v \in C \text{ o } u \in C \text{ y } v \in C_0\}.$$

Veamos los casos.

Caso 1: Sea $u = 0001$, Note que $|uw| \geq 5$, y las posibles longitudes para $v \in C$ son menores o iguales a 4, luego para este u, las ecuaciones no tienen solución.

Caso 2: Sea $u = 001$, por el argumento de las longitudes la única ecuación que se podría considerar es

$$001w = 0001.$$

Pero, el tercer símbolo de izquierda a derecha de la palabra de la izquierda es 1, mientras que el de la derecha es 0, por lo que ninguna ecuación tiene solución.

Caso 3: Sea $u = 01$, nuevamente por el argumento de las longitudes las ecuaciones que valen la pena considerar son

$$01w = \begin{cases} 001 \\ 0001 \end{cases}$$

Nuevamente el segundo símbolo de izquierda a derecha de la palabra de la izquierda es 1, mientras que en ambas palabras de la derecha el símbolo es 0, así para estas ecuaciones nuevamente no tenemos soluciones.

Caso 4: Para este último caso, siguiendo la misma idea de las longitudes, cuando $u = 1$ basta con considerar las siguientes ecuaciones

$$1w = \begin{cases} 01 \\ 001 \\ 0001 \end{cases}$$

Pero la palabra de la izquierda empieza en 1, mientras que las de la derecha en 0, por lo que nuevamente no hay soluciones.

Como en ningún caso existe w que cumpla las ecuaciones, tenemos que $C_1 = \varnothing$. Luego tenemos que $C_2 = C_3 = \cdots = \varnothing$. De esta manera tenemos que

$$C_\infty = \bigcup_{i=1}^{\infty} C_i = \varnothing.$$

Luego es claro que

$$C \cap C_\infty = \varnothing.$$

Así por el teorema de Sardinas-Patterson podemos concluir que C es unívocamente decodificable.

De esta manera concluimos por medio de Sardinas-Patterson que los C_1 y C_2, presentados en el enunciado, son unívocamente decodificables.

$\cup_{\vee}\cup$

Ejercicio 5
Pruebe, diseñando un algoritmo (sin usar el algoritmo de SardinasPaterson), que el código triario $\{aa, b, ba, abc\}$ es univocamente decodificable

Solución. El algoritmo para mostrar que el código es univocamente decodificable consiste en los siguientes pasos

1. Puesto que el código dado no es un código prefijo, intentaremos mirar si el código al "voltearlo" lo es por lo cual, el primer paso es voltear la palabras de la siguiente manera

$$C_1 = \{aa, b, ab, cba\}$$

2. Verificamos que C_1 es un código prefijo, por lo cual es univocamente decodificable.

3. Luego tenemos que $\{aa, b, ba, abc\}$ es univocamente decodificable.

Nuestro último paso se justifica en que si consideramos un código $C = x_1, \ldots, x_n$, al cual al voltearlo queda como $C_1 := x_1^v, \ldots, x_n^v$ es decodificable de manera única.
Tenemos que, C también es decodificable de manera única. Esto se debe a que $w = x_{i_1} \ldots x_{i_n}$ si y sólo si $w^R = x_{i_n}^R \ldots x_{i_1}^R$.

Ejercicio 6
Determine si el código triario $C = \{ab, cb, abbc, cbc, abb\}$ es unívocamente decodificable. Existe

un código binario instantáneo con las longitudes de palabras de C ? Si existe construyalo.

Solución. Procedamos por medio del algoritmo. Al igual que hicimos el el punto 4 cuando consideremos ecuaciones, solo tendremos en cuenta los casos con longitudes de palabra que al menos puedan coincidir. Definimos $C_0 = C$, ahora por definición

$$C_1 = \{w \in D^+ | \, uw = v, \text{ con } u \in C_0 \text{ y } v \in C \text{ o } u \in C \text{ y } v \in C_0\}.$$

Por lo que tenemos los siguientes casos:

- Si $u = ab$, las ecuaciones con longitudes de palabra coherente son

$$abw = \begin{cases} abbc \\ cbc \\ abb \end{cases}$$

  Luego, las posibles soluciones para w son bc en la primera y b en la última, en la segunda ecuación, la palabra de la izquierda empieza por a mientras que la derecha por c, así concluimos que no hay solución.

- Si $u = cb$, por las longitudes de palabra consideramos solamente

$$cbw = \begin{cases} abbc \\ cbc \\ abb \end{cases}$$

  Note que la primera ecuación y la última no tienen solución ya que la palabra de la izquierda empieza por c mientras la de la derecha por a. Pero, para la segunda si tomamos $w = c$, cumplimos la ecuación

- Si $u = cbc$, la única ecuación por longitudes es $cbcw = abbc$, pero esta no tiene solución por que la palabra de la izquierda empieza por c, mientras la de la derecha empieza por a. En cambio si $u = abb$, la ecuación $abbw = abbc$ tiene como solución $w = c$.

Así con todos los casos inspeccionados tenemos que $C_1 = \{b, c, bc\}$. Ahora tenemos que

$$C_2 = \{w \in D^+ | \, uw = v, \text{ con } u \in C_1 \text{ y } v \in C \text{ o } u \in C \text{ y } v \in C_1\}.$$

Note que por las longitudes de palabra las ecuaciones de la forma $uw = v$, donde $u \in C$ y $v \in C_1$, ya que $|uw| \geq 3$ mientras que $|v| \leq 2$. Así que solo observaremos cuando $u \in C_1$ y $c \in C$.

- Para los casos donde $u = b$ y $u = bc$, la palabra uw empieza por b, mientras que todas las palabras del código C empiezan por a o c, así ninguna ecuación tiene solución en este caso.

- Si $u = c$, basta considerar las ecuaciones que empiezan por ese símbolo, es decir, $cw = cb$ y $cw = cbc$. Luego $w = b$ o $w = bc$.

De esta manera concluimos que $C_2 = \{b, bc\}$. Note ahora que para C_3 podemos argumentar de manera muy similar a C_2. Para el caso donde $u \in C_2$ y $v \in C$, $u = b$ o $u = bc$, luego uw empieza por b, pero ninguna palabra en C empieza por b, así no hay soluciones. En caso de que $u \in C$ y $v \in C_2$, tenemos que $|uw| \geq 3$, mientras que $|v| \leq 2$. Así concluimos que $C_3 = C_4 = \cdots = \varnothing$. Luego

$$C_\infty = \bigcup_{i=1}^{\infty} C_i = C_1 \cup C_2 = \{b, c, bc\}.$$

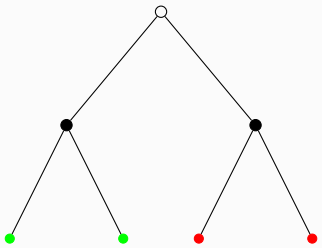
De esta manera $C \cap C_\infty = \varnothing$, y por el teorema de Sardinas-Patterson tenemos que C es unívocamente decodificable.

Ahora queremos ver si existe un código binario, es decir, $D = 2$, con longitudes de palabras $\ell_1 = 2, \ell_2 = 2, \ell_3 = 3, \ell_4 = 3$ y $\ell_5 = 4$, que sea instantáneo, esto lo podemos ver por medio de la desigualdad de Kraft

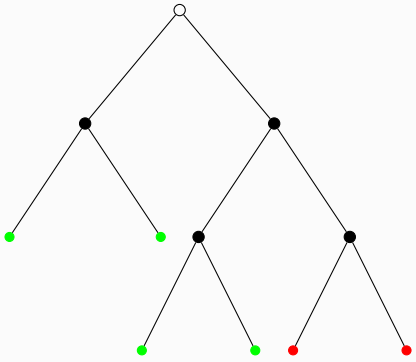
$$\begin{aligned}\sum_{i=1}^{5} D^{-\ell_i} &= 2^{-2} + 2^{-2} + 2^{-3} + +2^{-3} + +2^{-4} \\ &= 2^{-4}(2^2 + 2^2 + 2 + 2 + 1) \\ &\frac{4+4+2+2+1}{16} \\ &= \frac{13}{16} \leq 1\end{aligned}$$

Luego tal código si existe, por lo tanto podemos construirlo.

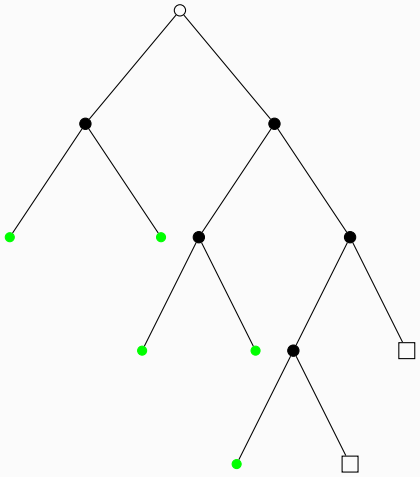
Primero realizamos un árbol de altura 2. Donde los nodos verdes serán hojas usadas mientras los rojos serán nodos terminales que usaremos de padres o se convertirán al final en hojas no usadas.



Luego extendemos a altura 3 por medio de los nodos terminales



Por último como solo queremos una palabra de longitud 4, el árbol queda de la siguiente manera



De esta manera el código instantáneo con las longitudes de palabra indicadas lo podemos dar, asignando a las aristas de la izquierda 0 y las de la derecha 1, tal que

$$C = \{00, 01, 100, 101, 1100\}.$$

∪˙∪

Ejercicio 7
Una fuente genera símbolos con una distribución de probabilidad

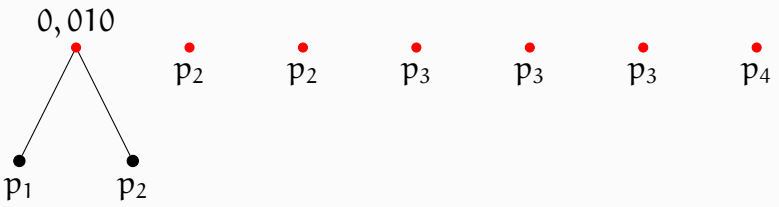
$$\{0{,}729, 0{,}081, 0{,}081, 0{,}081, 0{,}009, 0{,}009, 0{,}009, 0{,}001\}.$$

Determine un código binario que permita calcular la codificación de los símbolos de la fuente con menor longitud promedio. Cálcule la eficiencia del código.

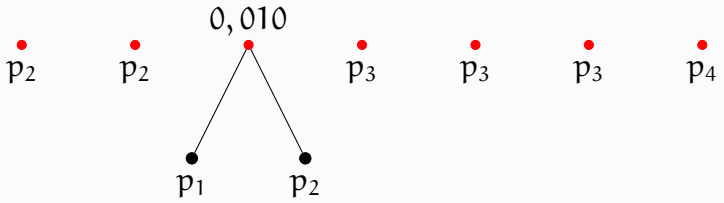
Solución. Como se nos pide hacerlo con la menor longitud promedio de palabra, podemos ir a lo seguro y aplicar el algoritmo de Huffman. Dado que $D = 2$, es decir, estamos trabajando en binario. Nuestro $r = 0$, por lo que no hay que añadir nodos nuevos y tenemos que el planteamiento inicial es

$$p_1 \quad p_2 \quad p_2 \quad p_2 \quad p_3 \quad p_3 \quad p_3 \quad p_4$$

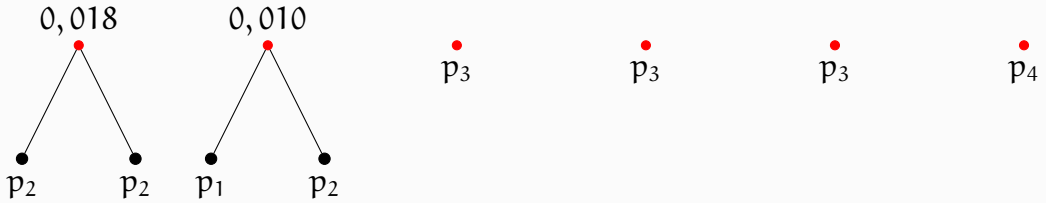
Donde $p_1 = 0,001, p_2 = 0,009, p_3 = 0,081$ y $p_4 = 0,729$, esto lo hacemos netamente por simpleza del árbol. Procedemos juntando los nodos de menor probabilidad por medio de un padre y sumamos sus probabilidades



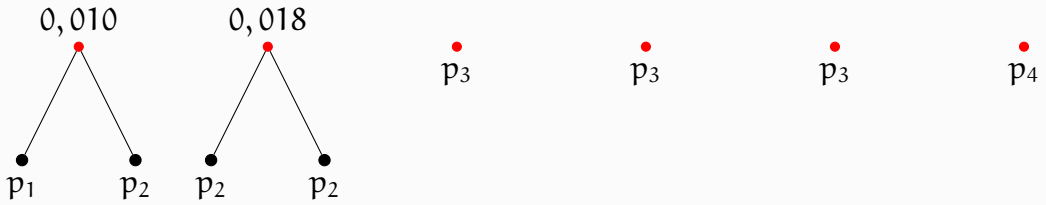
Luego reorganizamos nuevamente de orden menor a mayor



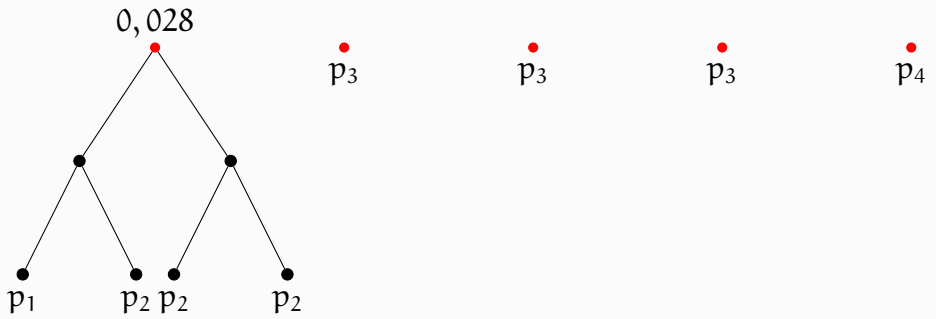
Volvemos a juntar los dos de menor probabilidad, es decir



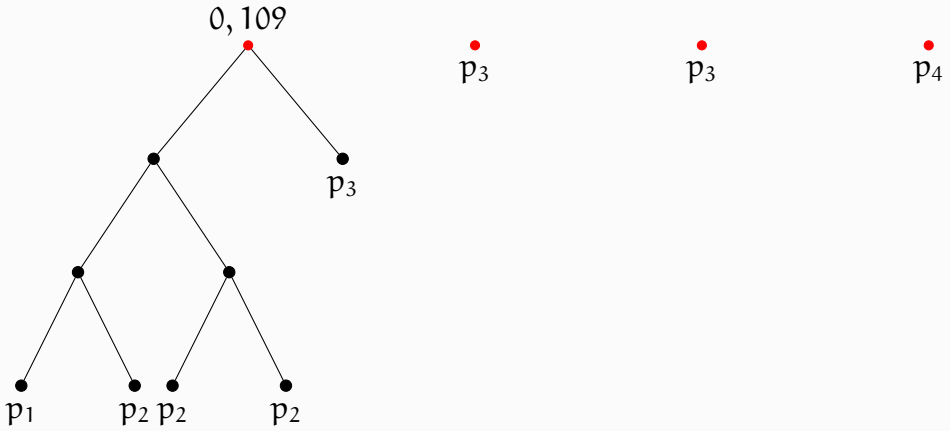
Nuevamente los organizamos



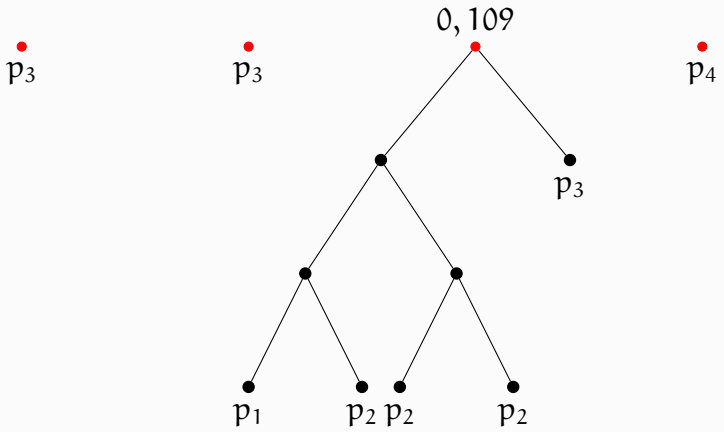
Ahora procedemos a unir los dos nodos de menor probabilidad



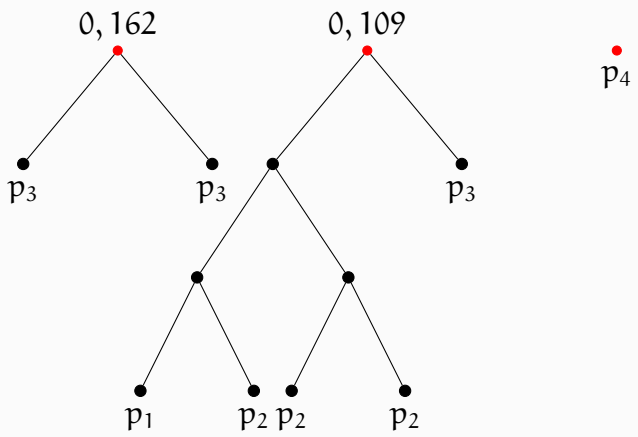
Note que ya esta organizado, asi que juntamos los dos nodos de menor probabilidad



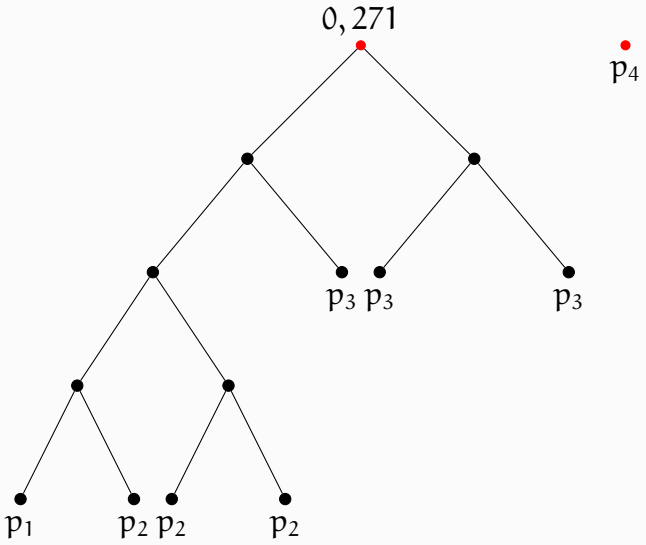
Note que ahora si tenemos que reordenar, luego



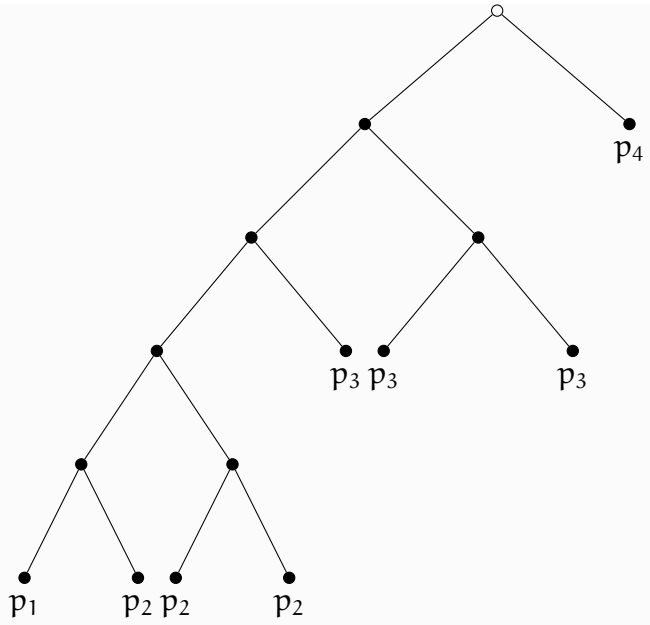
Procedemos a juntar los de menor probabilidad



Como no estan en orden, los vamos a organizar y de paso juntamos los de menor probabilidad, tal que



Luego como ya están en orden, simplemente creamos un nodo mas como raíz ya que solo quedan 2. Así el árbol de codificación es



Luego si asignamos a las ramas de la izquierda 0, y a las de la derecha 1 obtenemos el código C asociado a cada valor de la distribución de probabilidad tal que

0,729	0,081	0,081	0,081	0,009	0,009	0,009	0,001
1	001	010	011	00001	00010	00011	00000

Luego la longitud promedio de palabra viene dada por

$$\begin{aligned} L(C) &= \sum_{i=1}^{8} p_i |C(s_i)| \\ &= (0,001 + 0,009 + 0,009 + 0,009)5 + (0,081 + 0,081 + 0,081)3 + 0,729 \\ &= 0,14 + 0,729 + 0,729 \\ &= 1,598 \quad \text{bits/simbolo.} \end{aligned}$$

Mientras que la entropía de la fuente es

$$\begin{aligned} H(F) &= -\sum_{i=1}^{8} p_i \log_2(p_i) \\ &= -(0,001 \log_2(0,001) + 3 \cdot 0,009 \log_2(0,009) + 3 \cdot 0,081 \log_2(0,081) + 0,729 \log_2(0,729)) \\ &\approx 1,4069 \end{aligned}$$

Luego

$$\eta = \frac{H(F)}{L(C)} \approx 88\,\%$$

∪˳∪

Ejercicio 8
Diseñe un código de Fano para una fuente con la siguiente distribución de probabilidad

$$\{0{,}20, 0{,}19, 0{,}18, 0{,}17, 0{,}15, 0{,}10, 0{,}01\}$$

.

Solución. Este es el único ejercicio donde vamos a mostrar el proceso para crear un código de Fano, primero organizamos las palabras de menor probabilidad a mayor y les asignamos un nombre

A	B	C	D	E	F	G
0,01	0,10	0,15	0,17	0,18	0,19	0,20

Y aplicamos el algoritmo, de la siguiente manera

0.20	G	0.20	0	0			00
Paso 2 0.37	F	0.19	0	1	0		010
0.19 *Paso 4* 0.18	E	0.18	0	1	1		011
0.57 *Paso 1* 0.43	D	0.17	1	0			10
0.17 *Paso 3* 0.26	C	0.15	1	1	0		110
0.15 *Paso 5* 0.11	B	0.10	1	1	1	0	1110
0.10 *Paso 6* 0.01	A	0.01	1	1	1	1	1111

Ejercicio 9
Resuelva el ejercicio 1.5.1 de las notas de clase. Construya un código que no sea instantáneo cuyas longitudes de palabra cumplen la desigualdad de Kraft.

Solución. Sea $C = \{1, 10, 100, 1000\}$. Este código claramente no es instantáneo, ya que no es prefijo, por que la palabra 1 es segmento inicial de todas la demas palabras del código. Para este código tenemos que $D = 2$ ya que es un código binario, y tenemos longitudes de palabra $\ell_1 = 1, \ell_2 = 2, \ell_3 = 3$ y $\ell_4 = 4$. Así

$$\begin{aligned}\sum_{i=1}^{4} D^{-\ell_i} &= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} \\ &= 2^{-4}(2^3 + 2^2 + 2 + 1) \\ &= \frac{8+4+2+1}{16} \\ &= \frac{15}{16} \leq 1.\end{aligned}$$

De esta forma C es un código no instantáneo que cumple la desigualdad de Kraft como queriamos. ᗜˬᗜ

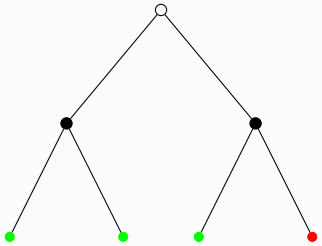
Ejercicio 10
Resuelva el ejercicio 1.5.2 de la notas de clase, calcule la eficiencia. Construya, en caso de que exista,

un código binario instantáneo constituido por 5 palabras de longitudes: 2, 2, 2, 3 y 4.

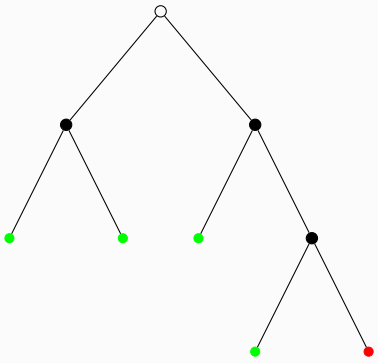
Solución. Primero verifiquemos que tal código exista. Tenemos que $D = 2$, por que es binario y tenemos longitudes de palabra $\ell_1 = 2, \ell_2 = 2, \ell_3 = 2, \ell_4 = 3$ y $\ell_5 = 4$. Asi

$$\begin{aligned}\sum_{i=1}^{5} D^{-\ell_i} &= 2^{-2} + 2^{-2} + 2^{-2} + 2^{-3} + 2^{-4} \\ &= 2^{-4}(2^2 + 2^2 + 2^2 + 2 + 1) \\ &= \frac{4+4+4+2+1}{16} \\ &= \frac{15}{16} \leq 1.\end{aligned}$$

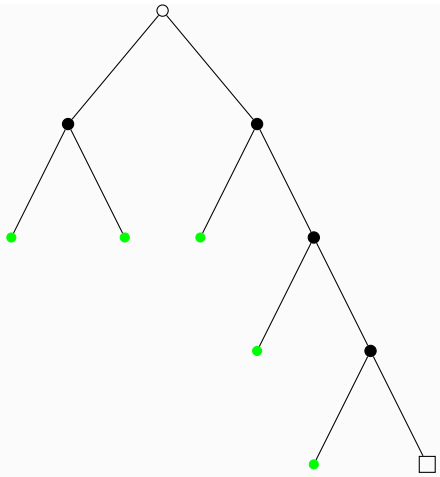
Como cumple la desigualdad de Kraft, sabemos que existe un código instantáneo (no completo ya que no se tiene la igualdad), con las longitudes de palabra indicadas. Siguiendo el algoritmo de construcción, tenemos para el paso uno el siguiente árbol de altura 2.



Donde los nodos verdes serán hojas, y el rojo es un nodo terminal. Para el segundo paso aumentamos a un árbol de altura 3 como se muestra a continuación



Para finalizar aumentamos el árbol a una altura 4



De esta forma si a las ramas que abren hacia la izquierda les asignamos el 0 y a las otras el 1, obtenemos el siguiente código instantáneo con las longitudes de palabra deseadas.

$$C = \{00, 01, 10, 110, 1110\}.$$

∪‿∪

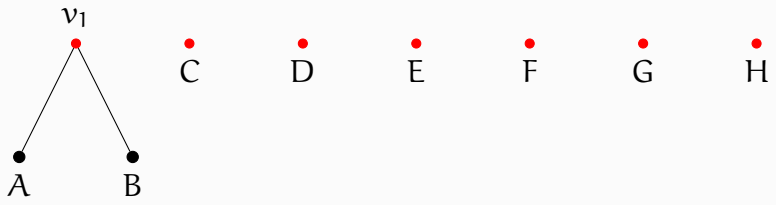
Ejercicio 11
Resuelva el ejercicio 2.3.1 de la página 36 de las notas de clase, calcule la eficiencia en cada caso. Considere el alfabeto S con la distribución de probabilidades que se muestra en la tabla:

A	B	C	D	E	F	G	H
0,02	0,03	0,04	0,04	0,12	0,20	0,20	0,35

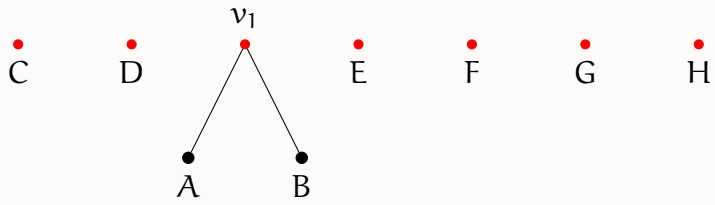
Suponga que se ha usado un código de Huffman para codificar los mensajes sobre un alfabeto binario. Si en el árbol de codificación se le asigna 1 a las ramas sobre la izquierda y 0 a las ramas sobre la derecha, ¿qué palabra representa la secuencia 111011111101110? Determine la longitud promedio de palabra para este código. ¿Cuál sería la codificación de los símbolos sobre un código triario?

Solución. Primero procederemos para el caso binario por medio del algoritmo de Huffman. Pintemos los respectivos 8 nodos en orden de menor a mayor según la distribución de probabilidad.

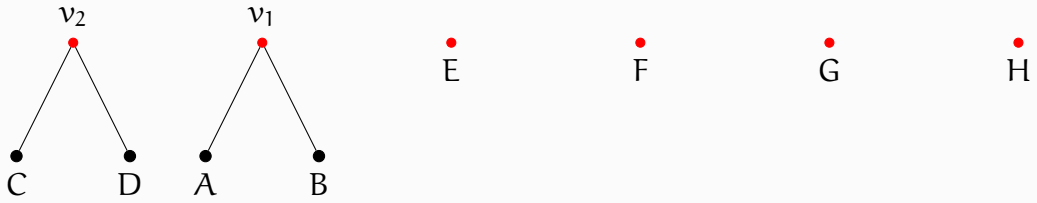
Donde el nodo rojo significa que está disponible. Como estamos en el caso binario no tenemos que agregar nodos nuevos al inicio y podemos proceder a crear un nodo padre para los dos de menos probabilidad, que en este caso son A y B. Quedando como nodos activos



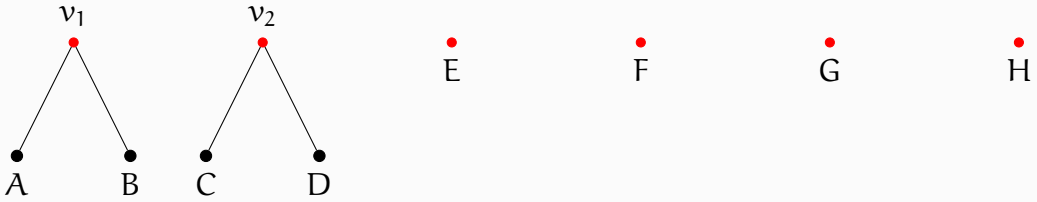
Ahora v_1 queda con las probabilidades sumadas de A y B, es decir $0,02 + 0,03 = 0,05$. Por el algoritmo tenemos que reorganizar los nodos tal que queden nuevamente en orden de probabilidades, tal que



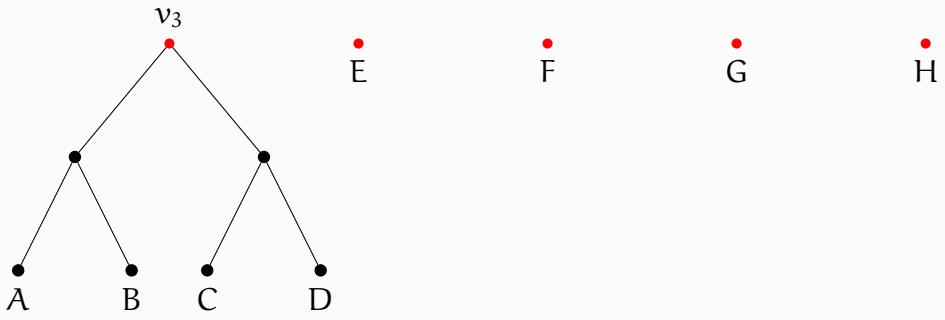
Ahora creamos un nodo padre para los dos de menor probabilidad, es decir C y D, a este lo llamaremos v_2 y tendra las probabilidad sumada, es decir $0,04 + 0,04 = 0,08$.



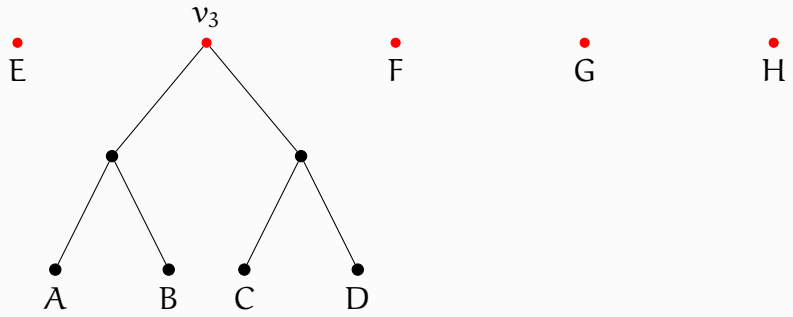
Nuevamente hay que reorganizar los nodos ya que la probabilidad de v_1 es menor a la de v_2.



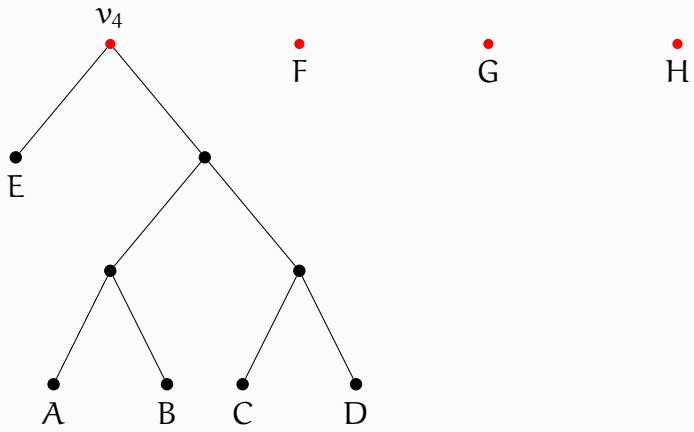
Y siguiendo el algoritmo ponemos un padre a v_1 y v_2, con probabilidad $0,05 + 0,08 = 0,13$.



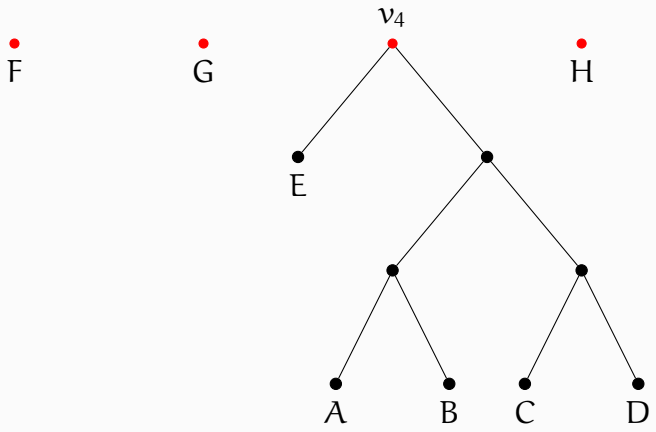
Ahora como la probabilidad de v_3 es mayor a la de E hay que cambiar el orden.



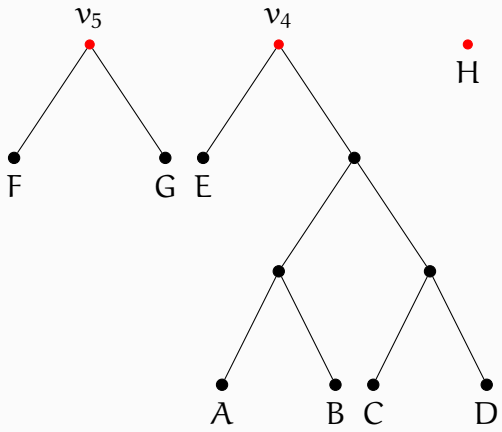
Ahora juntamos los nodos E y v_3 y les damos un padre con probabilidad de $0,12 + 0,13 = 0,25$.



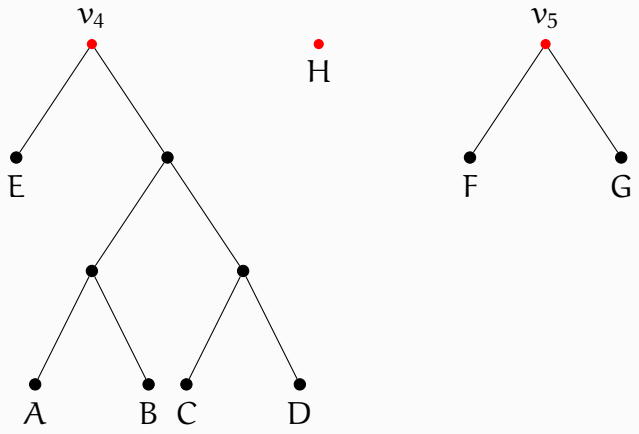
Luego la probabilidad de F y G es menor a la de v_4, asi reordenando



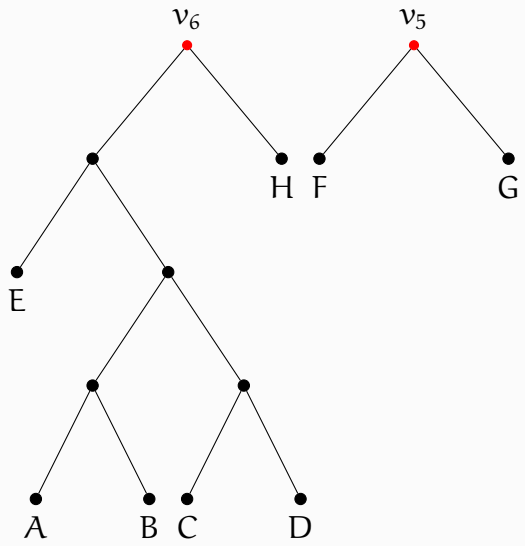
De esta manera tenemos que juntar los nodos F y G con un nuevo nodo padre, de probabilidad $0,20 + 0,20 = 0,40$.



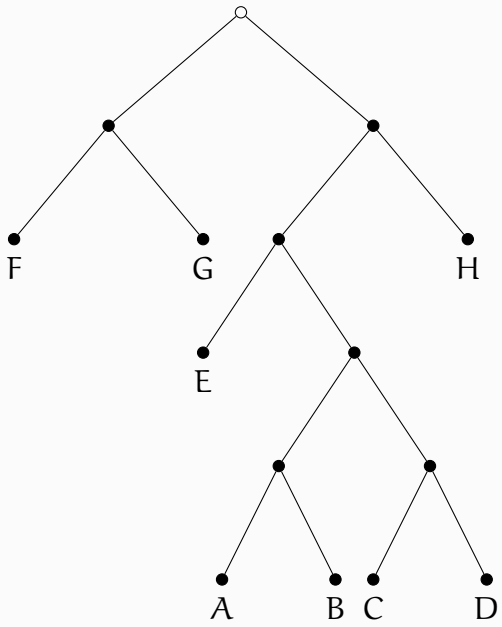
Nuevamente tenemos que ordenar, ya que la probabilidad de v_5 es mayor a la de v_4 y de H, asi



Para los nodos v_4 y H, asignamos un padre con probabilidad de $0,25 + 0,35 = 0,65$.



Ahora solo hay dos nodos, reorganizando y creando un último nodo activo llegamos a que el árbol de codificación es



Luego, las ramas de la izquierda las enumeramos con 1 y las de la derecha con 0 obtenemos

$$\begin{aligned}
C_2(F) &= 11\\
C_2(G) &= 10\\
C_2(H) &= 00\\
C_2(E) &= 011\\
C_2(A) &= 01011\\
C_2(B) &= 01010\\
C_2(C) &= 01001\\
C_2(D) &= 01000
\end{aligned}$$

Lo llamamos así debido a que es binario. Primero ya con la codificación, podemos determinar que palabra representa la secuencia dada en el enunciado. Si la separamos según la lectura sobre el árbol con comas nos damos cuenta que

$$11, 10, 11, 11, 11, 011, 10 = C_2(F)C_2(G)C_2(F)C_2(F)C_2(F)C_2(E)C_2(G).$$

Así esa secuencia representa la palabra FGFFFEG.

Luego, tenemos longitudes de palabra $2, 2, 2, 3, 5, 5, 5, 5$. Por lo tanto la longitud promedio de palabra es

$$\begin{aligned}
L(C_2) &= (0,20 + 0,20 + 0,35)2 + 0,12 \cdot 3 + (0,02 + 0,03 + 0,04 + 0,04)5\\
&= 2,51 \quad \text{bits/simbolo.}
\end{aligned}$$

Luego tomando la entropía de la fuente tenemos que

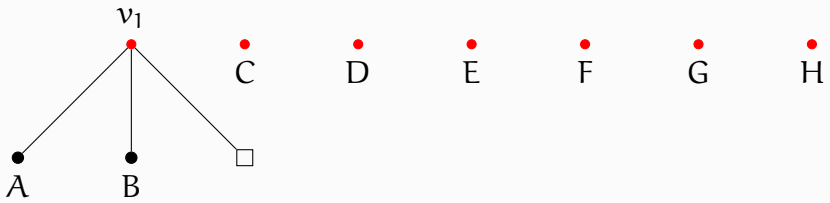
$$\begin{aligned}
H(F_2) &= -\sum_{i=1}^{8} p_i \log_2(p_i)\\
&= -(0,02\log_2(0,02) + 0,03\log_2(0,03) + 2 \cdot 0,04\log_2(0,04)\\
&\qquad + 0,12\log_2(0,12) + 2 \cdot 0,20\log_2(0,20) + 0,35\log_2(0,35))\\
&\approx 2,46.
\end{aligned}$$

Luego la eficiencia viene dada por

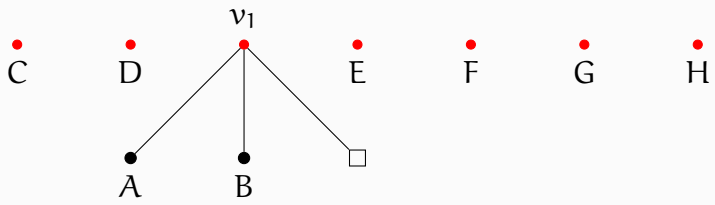
$$\eta_2 = \frac{H(F)}{L(C_2)} \approx \frac{2,46}{2,51} = 98\,\%.$$

Procedamos ahora con el algoritmo de Huffman pero para el caso triario. En este caso tenemos que $n - 1 = 8 - 1 = 7$ y que $D - 1 = 3 - 1 = 2$, luego el residuo de dividir 7 entre 2 es 1, por lo que en el proceso inicial hay que agregar un nodo nuevo.

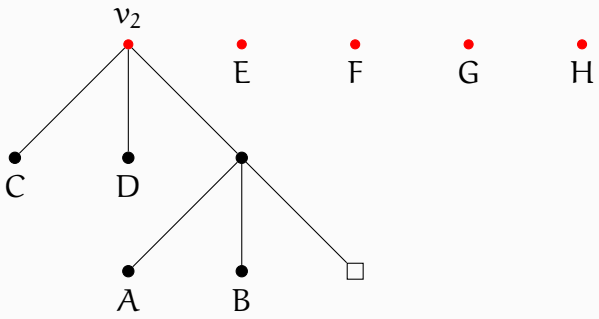
Procedemos igual que en el caso de binario, pero ahora hacemos un nuevo nodo padre v_1, para los nodos de menor probabilidad y el nodo nuevo que añadimos, tal que:



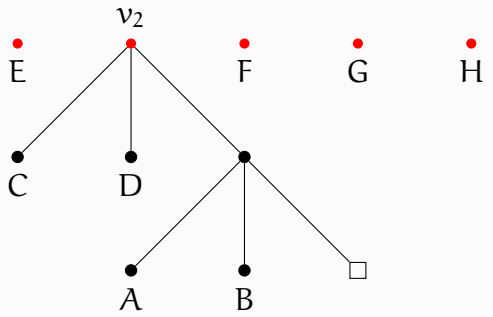
Donde la probabilidad de v_1 es la suma de la de los nodos hijos, es decir $0,05$. Luego toca reorganizar de acuerdo a las nuevas probabilidades



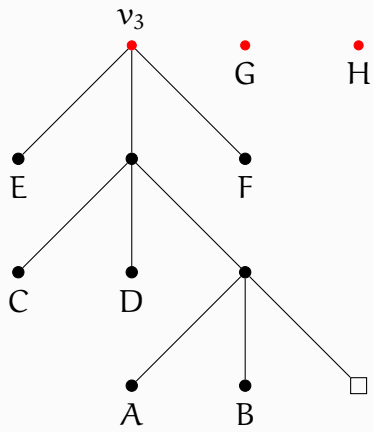
Ahora colocamos un nuevo nodo padre para los nodos C, D y v_1.



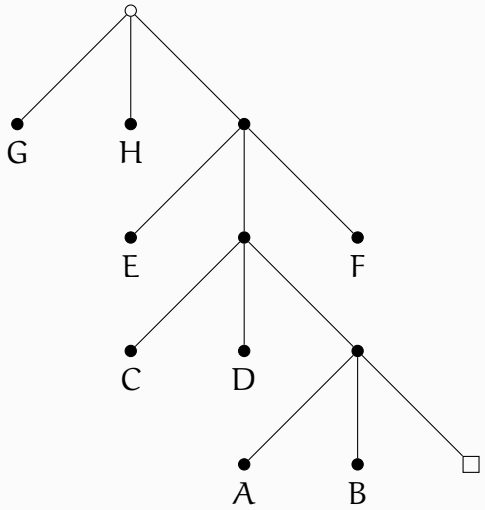
La probabilidad de este nuevo nodo es $0,04 + 0,04 + 0,05 = 0,13$. Por lo que nuevamente tenemos que organizar los nodos



Ahora le damos un padre a los nodos E, v_2 y F, que va a tener la probabilidad $0,12+0,013+0,20 = 0,45$.



Ahora como solo quedan tres nodos basta con ponerlos en orden de menor a mayor y juntarlos con el padre que sera la raíz, asi el árbol de codificación es



De esta manera si a cada rama de a izquierda a derecha le asignamos 0,1 y 2, respectivamente,

tenemos que la codificación de los símbolos es

$$\begin{aligned} C_3(G) &= 0, \\ C_3(H) &= 1, \\ C_3(E) &= 20, \\ C_3(F) &= 22, \\ C_3(C) &= 210, \\ C_3(D) &= 211, \\ C_3(A) &= 2120, \\ C_3(B) &= 2121. \end{aligned}$$

De manera abreviada en orden seria $C = \{2120, 2121, 210, 211, 20, 22, 0, 1\}$. Por hacer la comparativa hallemos la longitud promedio de palabra en este caso

$$\begin{aligned} L(C_3) &= 4 \cdot 0,02 + 4 \cdot 0,03 + 3 \cdot 0,04 + 3 \cdot 0,04 + 2 \cdot 0,12 + 2 \cdot 0,20 + 1 \cdot 0,20 + 1 \cdot 0,35 \\ &= 4 \cdot 0,05 + 3 \cdot 0,08 + 2 \cdot 0,32 + 0,55 \\ &= 1,63 \quad \texttt{tribits/simbolo}. \end{aligned}$$

Ahora para la entropía de la fuente en `tribits`, basta con dividir por $\log_2(3)$ a $H(F_2)$, ya que esto hace un cambio de base, así

$$H(F_3) \approx \frac{2,46}{\log_2(3)} \approx 1,55.$$

Luego tenemos que la eficiencia viene dada por

$$\eta_3 = \frac{H(F_3)}{L(C_3)} \approx \frac{1,55}{1,63} = 95\,\%.$$

∪˯∪

Ejercicio 12
Resuelva el ejercicio 3.5.1 de la página 61 de las notas de clase, calcule la eficiencia en cada caso. Considere una fuente que genera símbolos del alfabeto $S = \{0, 1\}$ con probabilidades $p(0) = 0{,}9$ y $p(1) = 0{,}1$. Diseñe códigos de Huffman y de Shannon-Fano para los alfabetos extendidos S^2, S^3 y S^4, en cada caso calcule la eficiencia.

Solución. Teniendo en cuenta que $S = \{0, 1\}$ tiene probabilidades $p(0) = 0.9$ y $p(1) = 0.1$, podemos decir que $S^2 = \{11, 10, 01, 00\}$ tiene probabilidades de $p^2 = \{0.01, 0.09, 0.09, 0.81\}$ respectivamente. Aplicando el algoritmo de Shannon-Fano obtenemos lo siguiente

00	0.81	0			0
01	0.09	1	0		10
10	0.09	1	1	0	110
11	0.01	1	1	1	111

Así, tenemos que $C = \{0, 10, 110, 111\}$ Luego tenemos que la longitud promedio de palabra es

$$\begin{aligned} L(C) &= (0,81) \cdot 1 + (0,09) \cdot 2 + (0,09) \cdot 3 + (0,01) \cdot 3 \\ &= 1{,}29 \text{ bits/símbolo.} \end{aligned}$$

Y la entropía para este caso es

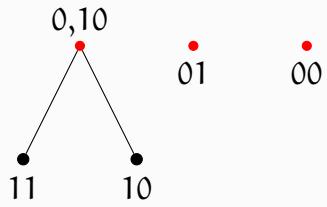
$$\begin{aligned} H(F^2) &= -(0,81)\log_2 0{,}81 - (0,09)\log_2 0{,}09 - (0,09)\log_2 0{,}09 - (0,01)\log_2 0{,}01 \\ &= 0{,}93799. \end{aligned}$$

Por lo tanto la eficiencia es

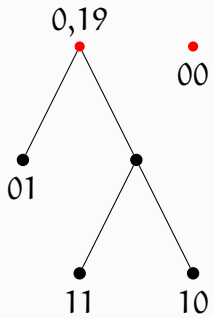
$$\eta = \frac{H(F^2)}{L(C)} \approx \frac{0{,}93799}{1{,}29} = 73\,\%.$$

Procedamos ahora por medio del algoritmo de Huffman. Organizando los nodos tenemos que

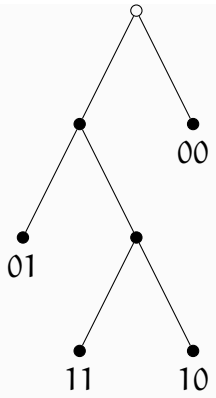
Note que como es binario, tenemos $r = 0$. Luego juntando los nodos de menor probabilidad



Ahora organizando en orden de menor a mayor y uniendo de paso los nodos de menor probabilidad, tenemos



Como ya estan ordenados, procedemos a crear el último nodo padre que sera la raíz, así obtenemos



Si a las ramas de la izquierda las etiquetamos con 0 y las de la derecha con 1, codificamos de menor a mayor longitud tal que

$$C = \{1, 00, 010, 011\}$$

Note que obtuvimos las mismas longitudes de palabra que en el caso de el código por medio de Shannon-Fano a pesar de ser una codificación distinta. Luego la eficiencia es la misma.
- Para el siguiente caso, tenemos que $S^3 = \{111, 110, 101, 011, 001, 010, 100, 000\}$ tenemos que las probabilidades de $p^3 = \{0.001, 0.009, 0.009, 0.009, 0.081, 0.081, 0.081, 0.729\}$ respectivamente. Aplicando el algoritmo de Shannon-Fano obtenemos

000	0.729	0					0
100	0.081	1	0	0			100
010	0.081	1	0	1			101
001	0.081	1	1	0			110
011	0.009	1	1	1	0	0	11100
101	0.009	1	1	1	0	1	11101
110	0.009	1	1	1	1	0	11110
000	0.001	1	1	1	1	1	11111

Así, tenemos que $C = \{0, 100, 101, 110, 11100, 11101, 11110, 11111\}$, por lo cual la longitud promedio de palabra es

$$\begin{aligned} L(C) =& 5(0{,}001) + 5(0{,}009) + 5(0{,}009) + 5(0{,}009) + 3(0{,}081) + 3(0{,}081) + 3(0{,}081) + 1(0{,}729) \\ =& 1{,}598 \text{ bits símbolo..} \end{aligned}$$

Y la entropía para este caso es

$$\begin{aligned} H(F^3) &= -(0{,}001)\log_2 0{,}001 - 3[(0{,}009)\log_2 0{,}009] - 3[(0{,}081)\log_2 0{,}081] - (0{,}729)\log_2 0{,}729 \\ &= 1{,}40698. \end{aligned}$$

Por lo tanto la eficiencia es

$$\eta = \frac{H(F^3)}{L(C)} = \frac{1{,}40698}{1{,}598} \approx 88\,\%.$$

Note que esta distribución de probabilidad es la misma dada en el Ejercicio 7, luego una codificación por medio de Huffman es la dada en el punto, tal que

$$C = \{1, 001, 010, 011, 00001, 00010, 00011, 00000\}.$$

Note que esta codificación tiene las mismas longitudes de palabra que por medio de Shannon-Fano, luego la eficiencia es la misma nuevamente.
- Por último, tenemos que

$$S^4 = \{1111, 1110, 1101, 1011, 0111, 1001, 1010, 1100, 0110, 0011, 0101, 1000, 0100, 0010, 0001, 0000\}$$

y así, las probabilidades de

$$p^4 = \{0.0001, 0.0009, 0.0009, 0.0009, 0.0009, 0.0081, 0.0081, 0.0081, 0.0081, 0.0081, 0.0081, 0.0729, 0.0729, 0.0729, 0.0729, 0{,}6561\}.$$

respectivamente. Aplicando el algoritmo de Shannon-Fano obtenemos

0000	0,6561	0									0
0001	0,0729	1	0	0							100
0010	0,0729	1	0	1							101
0100	0,0729	1	1	0							110
1000	0,0729	1	1	1	0						1110
0101	0,0081	1	1	1	1	0	0				111100
0011	0,0081	1	1	1	1	0	1	0			1111010
0110	0,0081	1	1	1	1	0	1	1			1111011
1100	0,0081	1	1	1	1	1	0	0			1111100
1010	0,0081	1	1	1	1	1	0	1			1111101
1001	0,0081	1	1	1	1	1	1	0			1111110
0111	0,0009	1	1	1	1	1	1	1	0	0	111111100
1011	0,0009	1	1	1	1	1	1	1	0	1	111111101
1101	0,0009	1	1	1	1	1	1	1	1	0	111111110
1110	0,0009	1	1	1	1	1	1	1	1	1	1111111110
1111	0,0001	1	1	1	1	1	1	1	1	1	1111111111

Así, tenemos que $C = \{0, 100, 101, 110, 1110, 111100, 1111010, 1111011, 1111100, 1111101, 1111110, 111111100, 111111101, 111111110, 1111111110, 1111111111\}$, por lo cual la longitud promedio de palabra es

$$\begin{aligned} L(C) =& 1(0{,}6561) + 3(3(0{,}0729)) + 4(0{,}0729) + 6(0{,}0081) + 5(7(0{,}0081)) + 3(9(0{,}0009)) \\ &+ 10(0{,}0009) + 10(0{,}0001) \end{aligned}$$

$$= 1{,}9702 \text{ bits/ símbolo.}$$

Y la entropía para este caso es

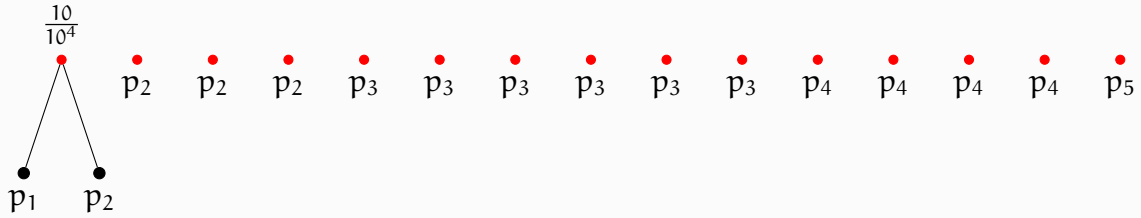
$$\begin{aligned} H(F^4) &= -0{,}6561 \cdot \log_2(0{,}6561) - 4(0{,}0729 \cdot \log_2(0{,}0729)) - 6(0{,}0081 \cdot \log_2(0{,}0081)) \\ &\quad - 4(0{,}0009 \cdot \log_2(0{,}0009)) - 0{,}0001 \cdot \log_2(0{,}0001) \\ &= 1{,}87598. \end{aligned}$$

Por lo tanto la eficiencia es

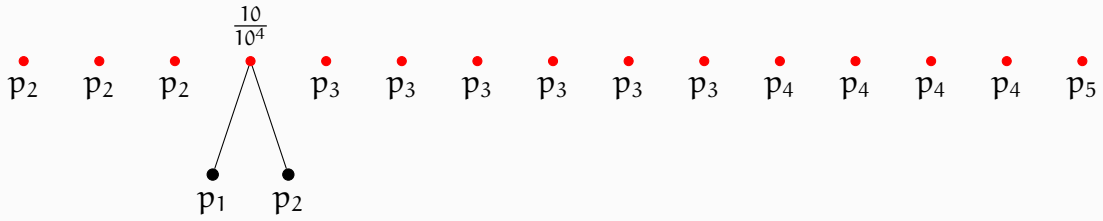
$$\eta = \frac{H(F^3)}{L(C)} = \frac{1{,}87598}{1{,}9702} \approx 95\,\%.$$

Ahora procedamos por medio de el algoritmo de Huffman. Tenemos primero 16 nodos distintos que organizaremos de menor a mayor probabilidad

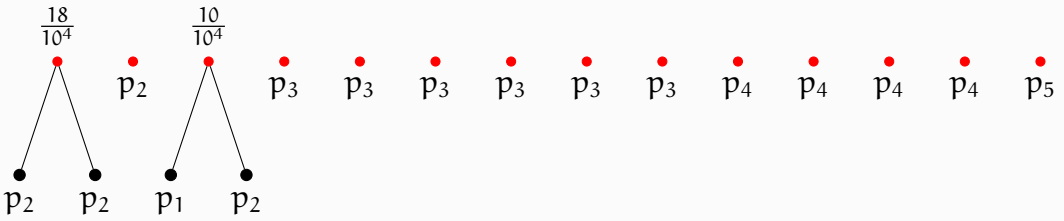
Donde $p_1 = 0,0001, p_2 = 0,0009, p_3 = 0,0081, p_4 = 0,0729$ y $p_5 = 0,6561$. Además, cada nodo marcado con alguna p_i, hace referencia a una palabra de S^4, esto es meramente por facilidad, pero a fin de cuentas la codificación sera equivalente. Ahora procedemos a unir los nodos de menor probabilidad



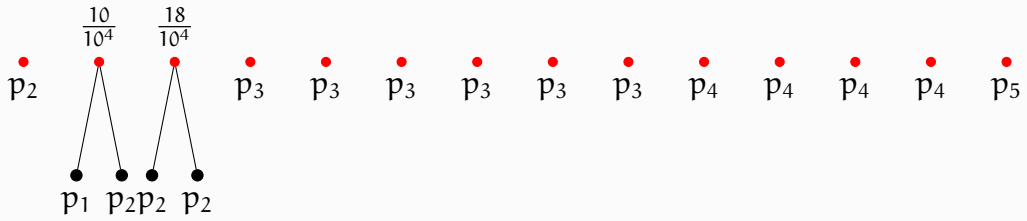
Ahora reorganizamos



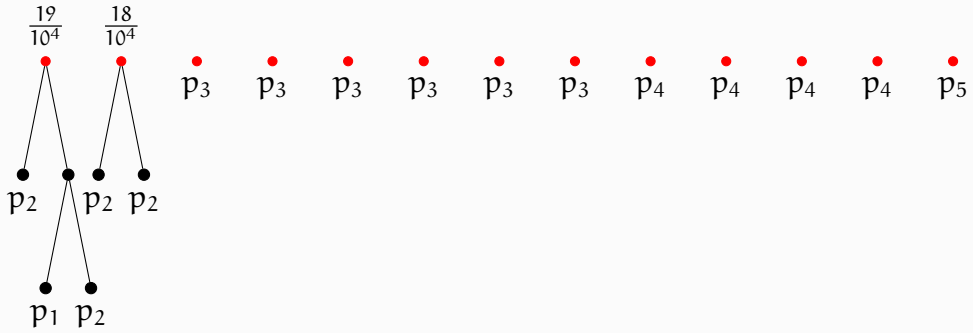
Juntamos nuevamente los de menor probabilidad



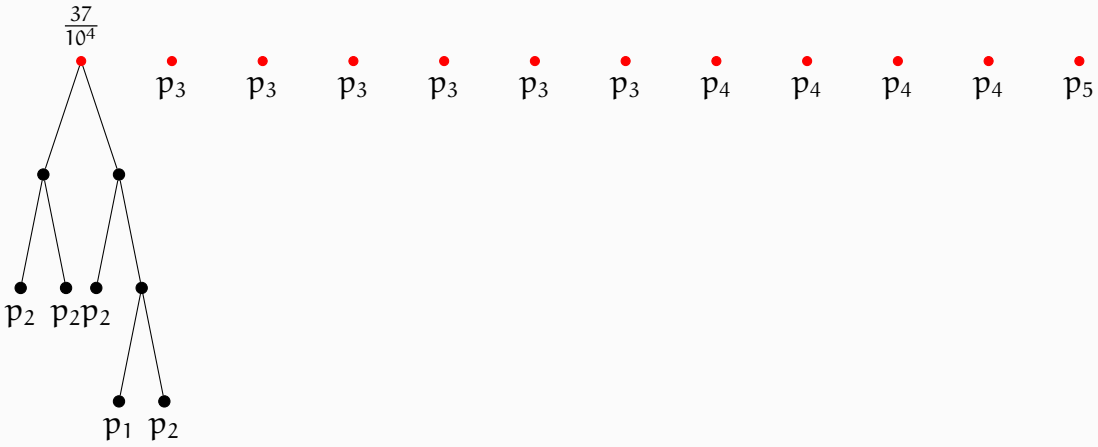
Luego lo ordenamos nuevamente



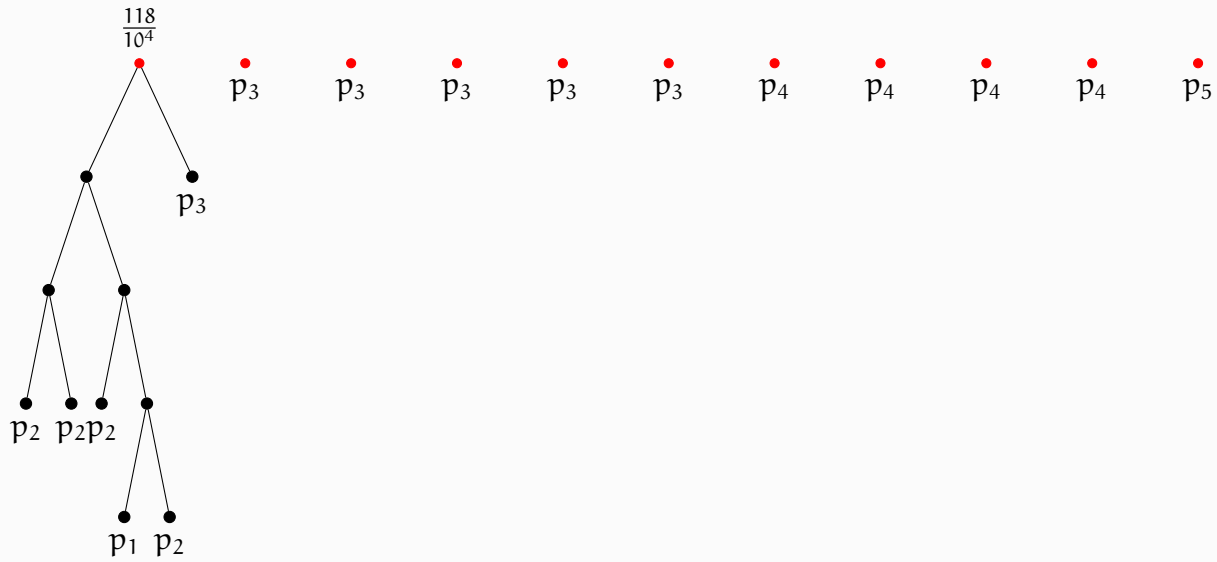
Ahora juntamos los de menor probabilidad



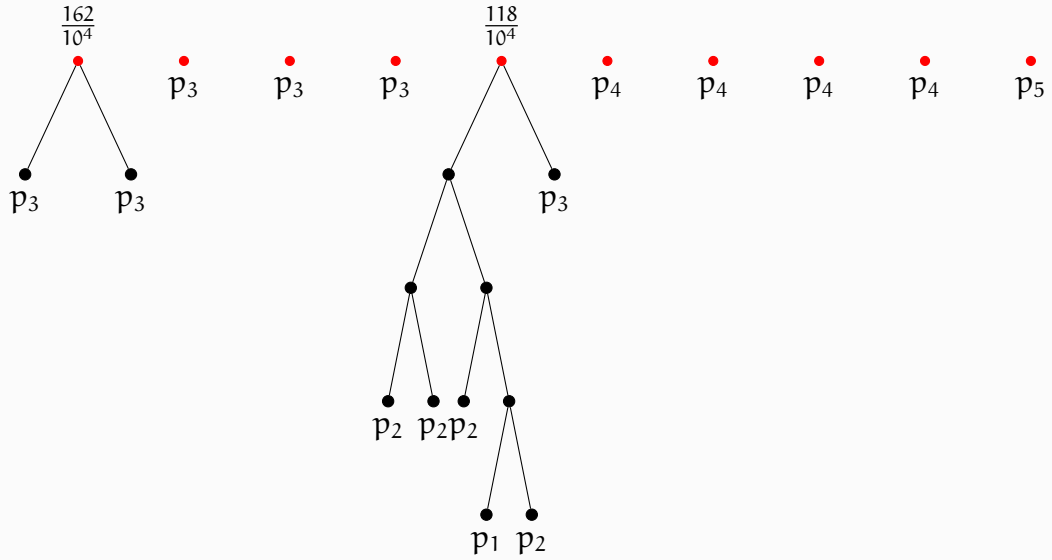
Si reorganizamos y juntamos los nodos de menor probabilidad obtenemos



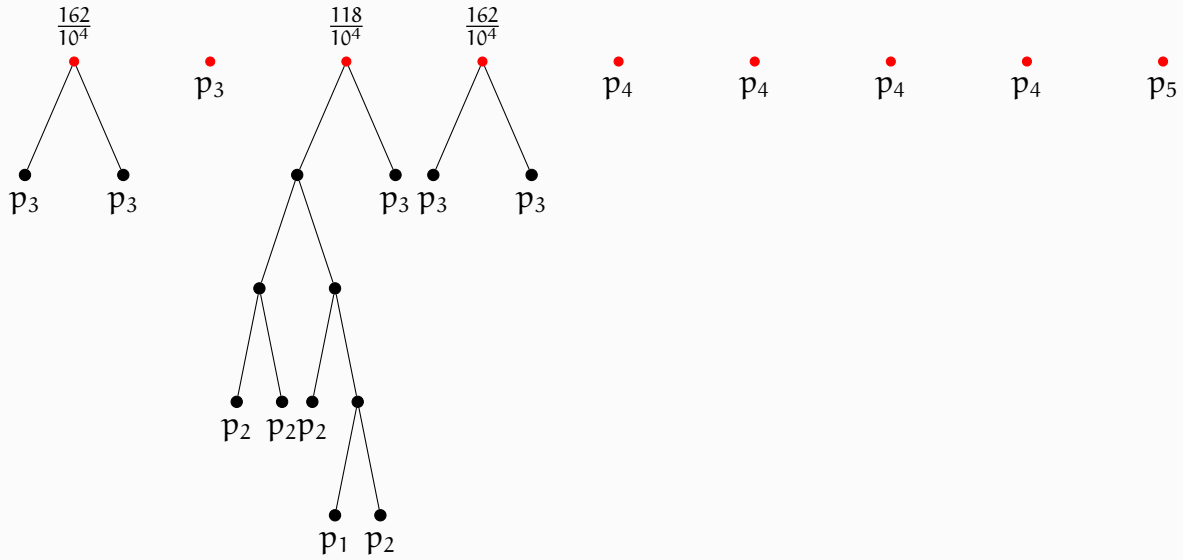
Como ya esta organizado, podemos juntar los nodos tal que



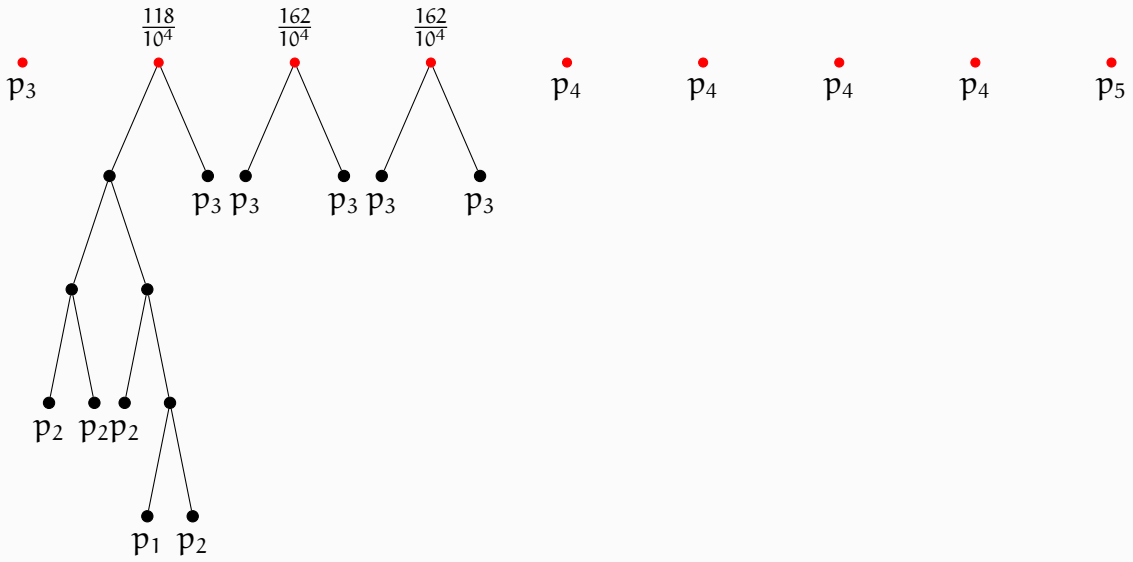
Organizando y juntando los de menor probabilidad tenemos



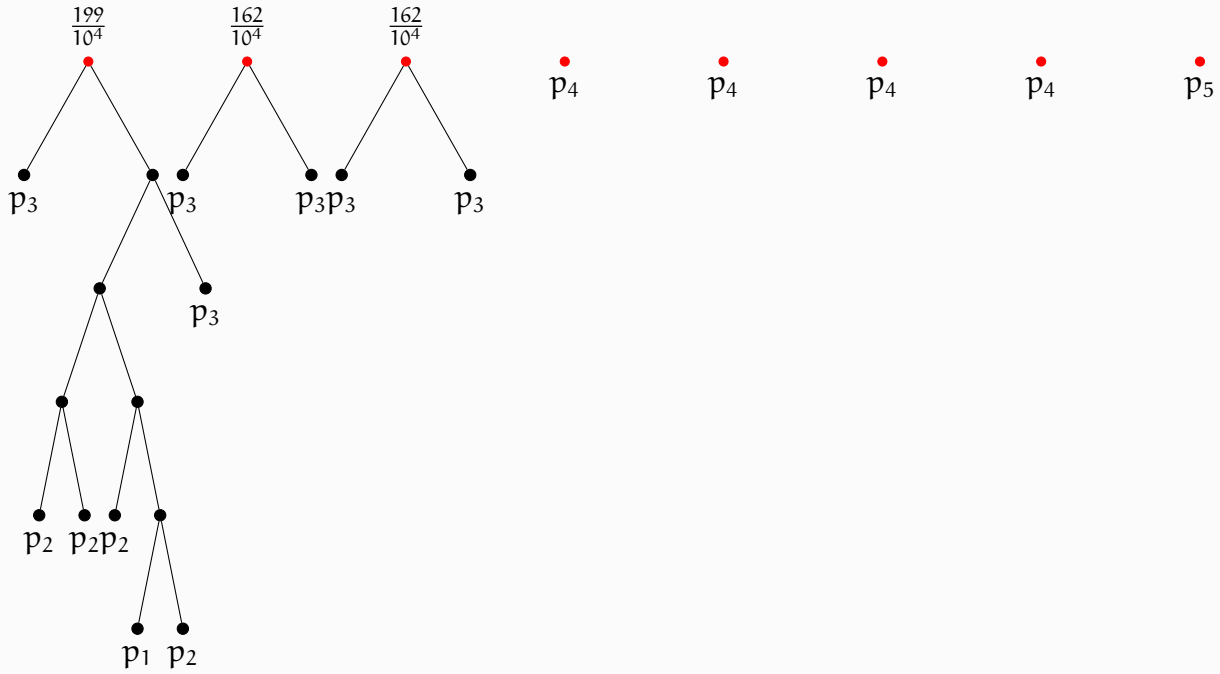
Nuevamente organizando y juntado los de menor probabilidad



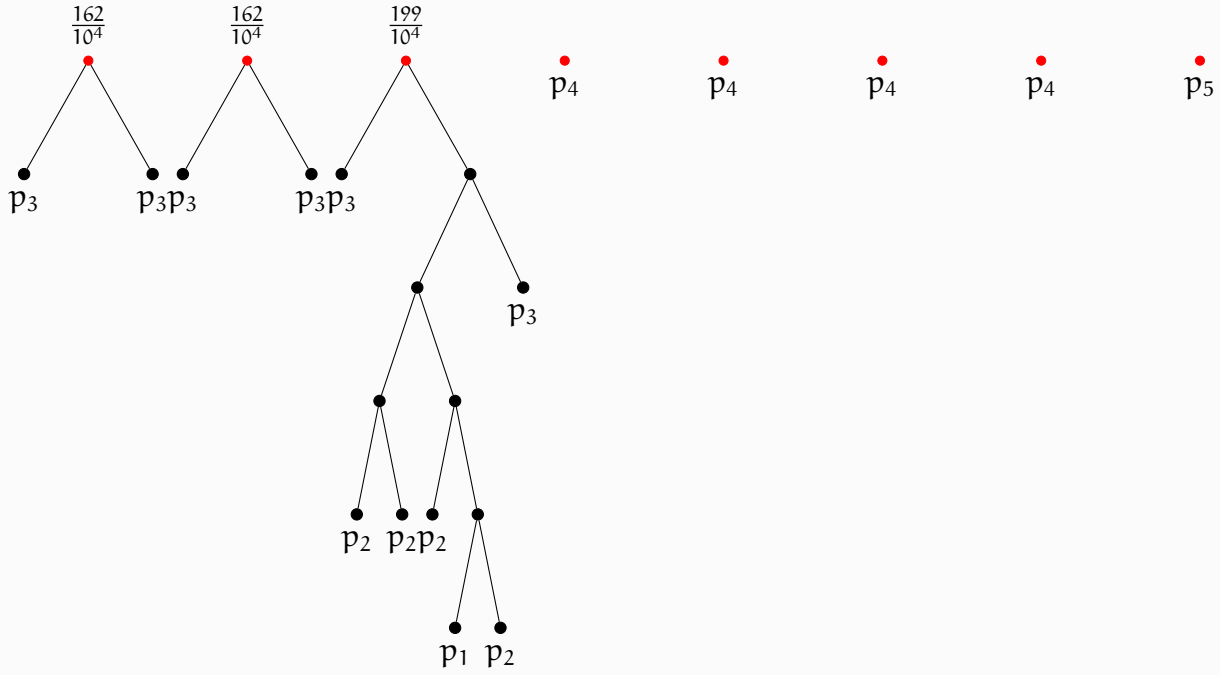
Reordenando



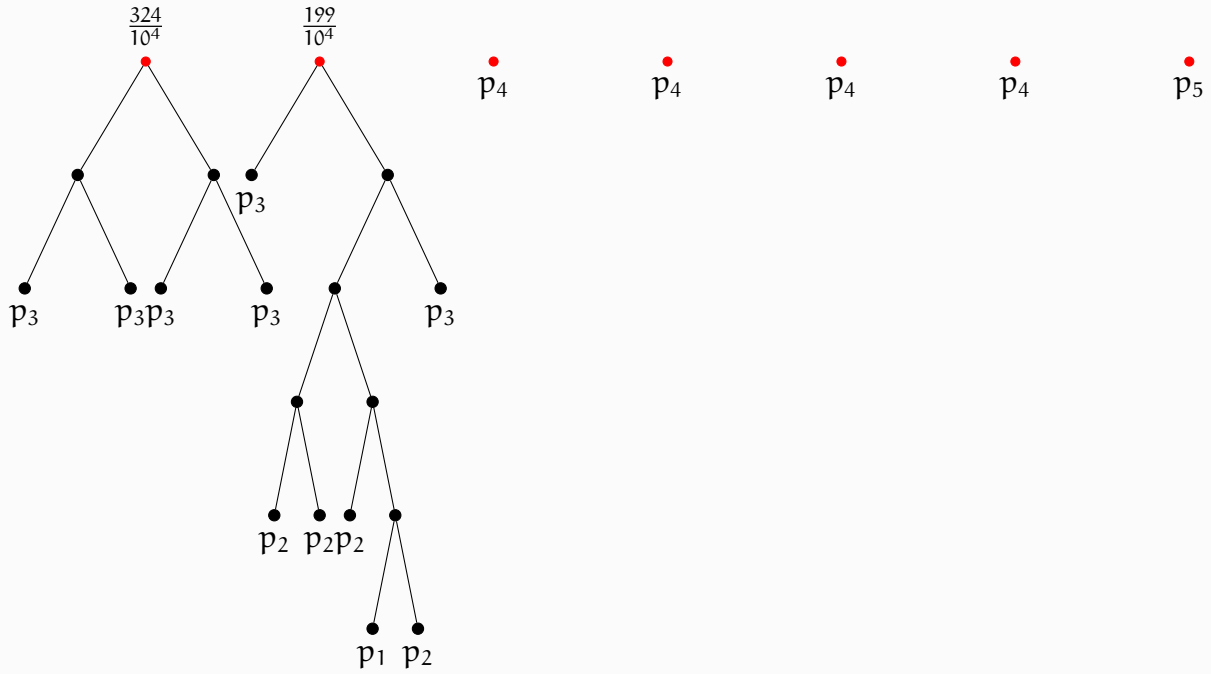
Uniendo los nodos de menor probabilidad obtenemos



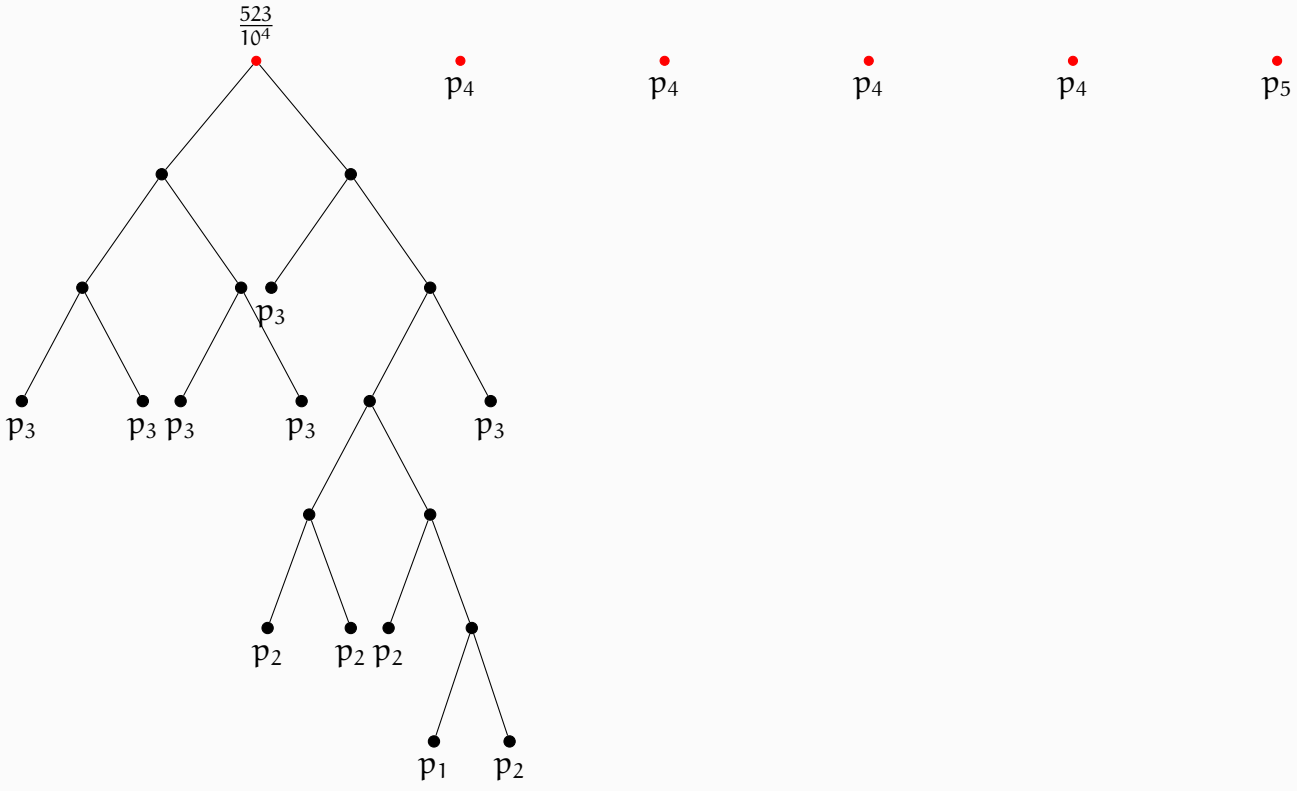
Nuevamente organizamos



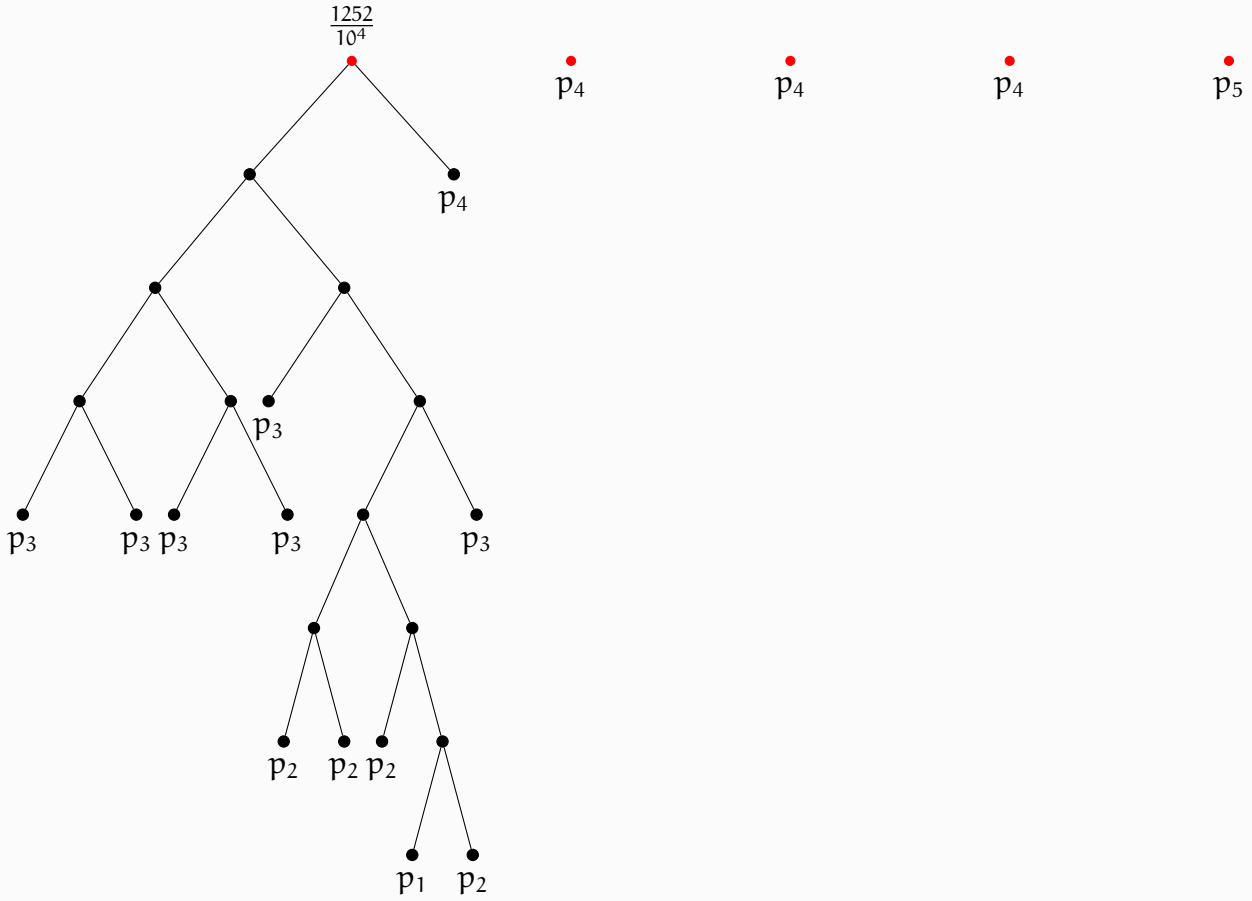
Unimos los dos nodos de menor probabilidad



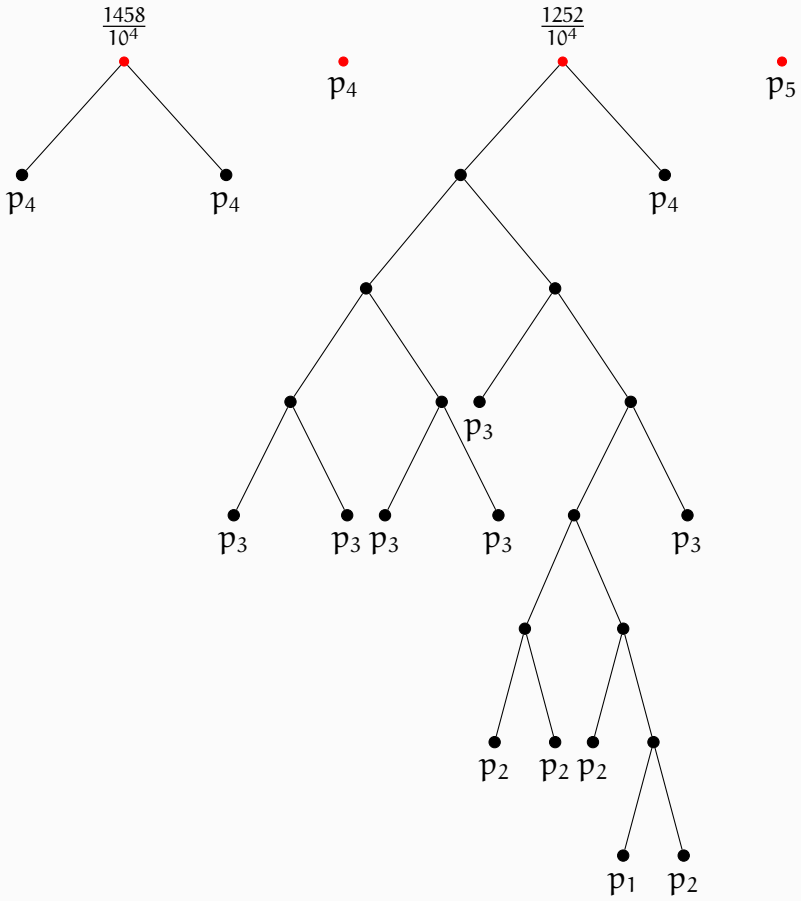
Por motivos de visualización no reordenaremos, ya que los dos nodos que uniremos son los que están al final, como se ve a continuación



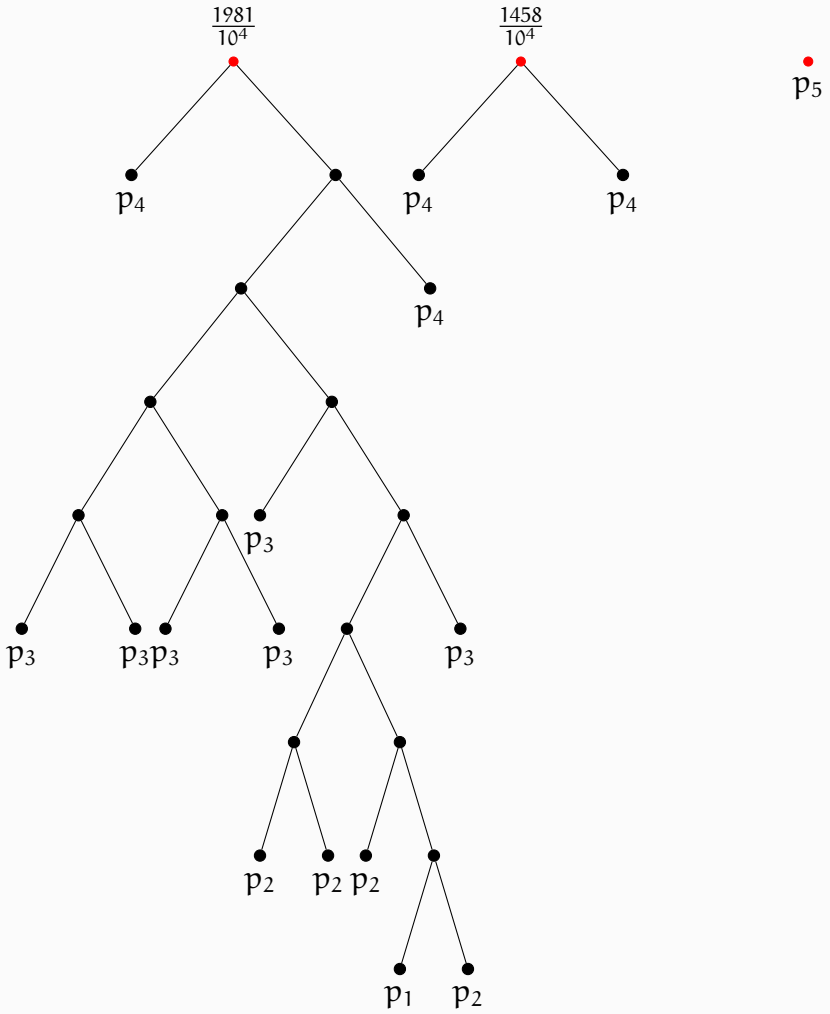
Note que ahora podemos juntar los dos primeros nodos, sin necesidad de reorganizar



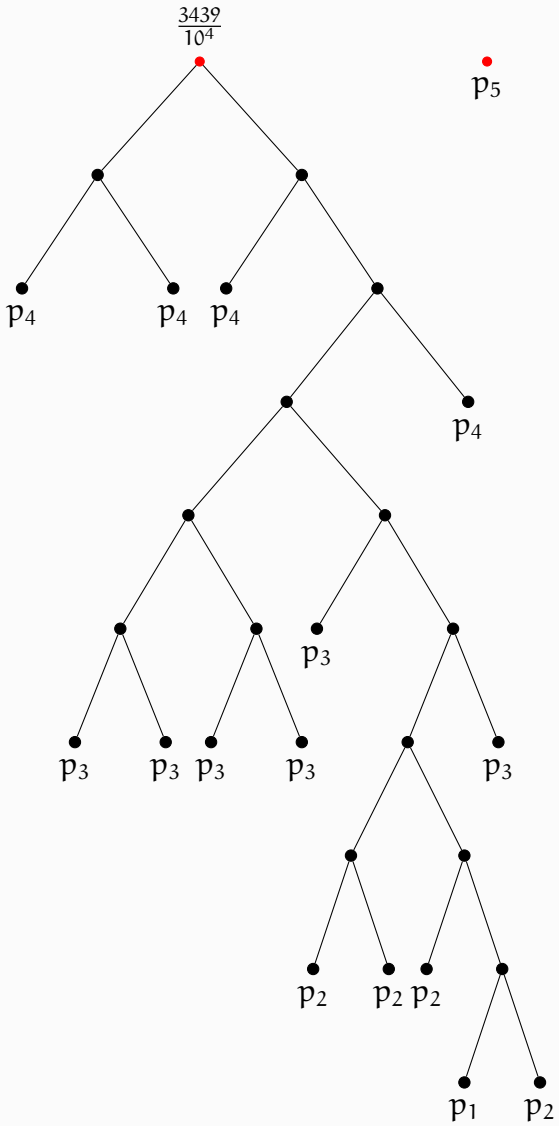
Reordenando y juntando los nodos de menor probabilidad



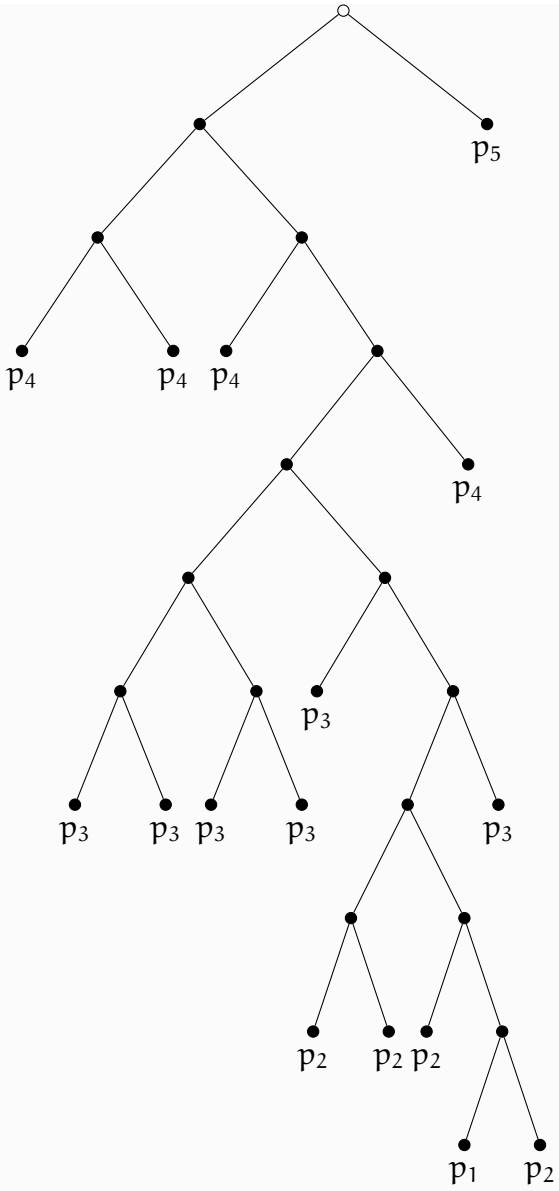
Nuevamente reorganizamos y juntamos los de menor probabilidad



Reordenando y uniendo los de menor probabilidad, tenemos



Note que como solo quedan dos nodos y ya están en orden, podemos finalizar colocando el nodo raíz



De esta manera si asignamos 0 a las ramas de la izquierda y 1 a las de la derecha, obtenemos de menor a mayor longitud

$$C = \{1, 000, 001, 010, 0111, 011010, 0110000, 0110001, 0110010, 0110011, 0110111, 011011000, 011011001, 011011010, 0110110110, 0110110111\}.$$

Note que las longitudes de palabra de este código, coinciden con las de el código realizado por Shannon-Fano, de esta manera tienen la misma eficiencia.