Auxiliar #2 - Cotas Inferiores

21 de septiembre de 2020 - Bernardo Subercaseaux

Problema 1. (★★) Demuestre que para determinar si un grafo no dirigido de n nodos tiene un nodo de grado 3 se requieren $\binom{n}{2} - n$ consultas de adyacencia.

Solución 1 . El adversario $\mathcal{E}$ mantendrá un grafo $G_{\mathcal{E}}$ de n nodos, que contiene las aristas (u, v) que no han sido *negadas* por $\mathcal{E}$. Inicialmente, dado que ninguna consulta ha sido negada, $G_{\mathcal{E}}$ es el grafo completo de n nodos y tiene $\binom{n}{2}$ aristas. A continuación se describe el proceso mediante el cual $\mathcal{E}$ responderá a la consultas. Mientras en $G_{\mathcal{E}}$ exista un nodo v de grado 3 o más, $\mathcal{E}$ negará cualquier consulta (u, v), quitando la arista (u, v) de el grafo $G_{\mathcal{E}}$ que mantiene. Si en cambio todo nodo en $G_{\mathcal{E}}$ tiene grado menor a 3, entonces $\mathcal{E}$ aceptará cualquier consulta (u, v).

La clave es que mientras $\mathcal{E}$ no haya comenzado a aceptar consultas, ningún algoritmo tiene suficiente información para concluir la respuesta; en efecto, mientras $\mathcal{E}$ no ha comenzado a aceptar consultas se las siguientes dos propiedades:

1. Existe un nodo $v \in G_{\mathcal{E}}$ de grado 3 o más.
2. Todas las aristas en $G_{\mathcal{E}}$ son aristas por las cuáles no se ha consultado, y por tanto todo subgrafo de $G_{\mathcal{E}}$ es consistente con la información dada hasta el momento.

Y por lo tanto si un algoritmo, durante esta fase, dice que el grafo no tiene un nodo de grado 3, el adversario puede mostrar el subgrafo de $G_{\mathcal{E}}$ en que v tiene grado exactamente 3, haciendo fallar al algoritmo. Por el contrario, si un algoritmo dice que el grafo sí tiene un nodo de grado 3, el adversario puede mostrar el subgrafo vacío de $G_{\mathcal{E}}$ (recordemos que ninguna consulta ha sido aceptada), haciendo fallar al algoritmo.

Solo resta por estudiar cuántas consultas debe recibir $\mathcal{E}$ hasta que comience a aceptar. Dado que la condición para aceptar es que todos los nodos en $G_{\mathcal{E}}$ tengan grado a lo más 2, en ese momento $G_{\mathcal{E}}$ no tiene más de n aristas (lema del apretón de manos). Puesto que (i) $G_{\mathcal{E}}$ comienza con $\binom{n}{2}$ aristas, (ii) ningún algoritmo puede dar una respuesta correcta hasta que $G_{\mathcal{E}}$ no tenga $\leq n$ aristas y (iii) en cada consulta se borra a lo más una arista de $G_{\mathcal{E}}$, podemos concluir que cualquier algoritmo correcto requiere al menos $\binom{n}{2} - n$ consultas para decidir correctamente.

Problema 2. (★★★) Demuestre que dados n elementos se requieren al menos $n + \lceil \lg n \rceil - 2$ comparaciones para identificar tanto el máximo como el segundo máximo entre ellos. Muestre que esta cota es ajustada.

Solución 2 . El apunte presenta una demostración en las páginas 16-18. Por completitud, aquí presento una versión simplificada.

En primer lugar, dado que se requiere determinar el máximo elemento, se requieren al menos las $n - 1$ comparaciones necesarias para esto (cota ya demostrada en clases). Sea g el elemento máximo, y m el número de elementos que fueron comparados con g por un algoritmo cualquiera, llamaremos a estos elementos *candidatos*. Notemos que cualquier elemento, excepto g, debe haber perdido una comparación durante el proceso de identificación del máximo (de lo contrario ningún algoritmo podría estar seguro de que no fuese mayor que g). Consecuentemente, el segundo máximo ha de ser uno de los candidatos, puesto que si no fuese candidato, habría perdido una comparación con un elemento h distinto de g, y tener dos elementos mayores

que él (g y h) contradeciría el que fuese el segundo máximo. Notemos además que las $n-1$ comparaciones con que se obtuvo g no permiten deducir nada con respecto a la relación entre los m candidatos; en efecto, lo único que sabe cualquier algoritmo es que cada uno de ellos perdió contra g. Por lo tanto, podemos aplicar nuevamente la cota conocida para obtener el máximo de un conjunto de elementos sin información previa, estableciendo que se requieren $m-1$ comparaciones entre los m candidatos. En total se requieren $n-1+m-1 = n+m-2$ comparaciones. Ahora mostraremos que un adversario puede forzar $m = \lceil \lg n \rceil$. El adversario mantiene como información adicional un arreglo $W[1..n]$, donde la posición $W[i]$ representa cuantos elementos son menores o iguales a i, a menos que i haya perdido una comparación, en cuyo caso se asignará $W[i] := 0$. Inicialmente $W[i] := 1$ para cada elemento i. En cada comparación entre elementos i y j, el adversario dirá que gana i si $W[i] \geq W[j]$ y j si no. Si gana i, el adversario asignará $W[i] := W[i] + W[j]$ y $W[j] := 0$. Para que un algoritmo sea capaz de determinar el máximo g, debe haber un elemento i tal que $W[i] = n$, puesto que el máximo es mayor o igual que n elementos. Notemos que inicialmente $W[i] = 1$, y que finalmente $W[i] = n$. Pero en cada comparación que gana i sobre j a $W[i]$ se le suma $W[j]$, que por definición cumple $W[j] \leq W[i]$, y por lo tanto $W[i]$ a lo más se duplica. Dado que (i) en cada comparación victoriosa $W[i]$ a lo más se duplica, (ii) inicialmente $W[i] = 1$ y (iii) finalmente $W[i] = n$, concluimos que el adversario fuerza al menos $\lceil \lg n \rceil$ comparaciones victoriosas para el máximo. De esto se deduce que $m \geq \lceil \lg n \rceil$, con lo que se concluye la cota deseada.