Tarea Integradora II

Juan José López López - A00381190
Juan David Patiño - A00381293
Juan Eduardo Yustes - A00380718

Computación y Estructuras Discretas I
Facultad de Ingeniería
Jeison Trujillo

Universidad ICESI
Cali, Colombia
2022

Fase 1. Identificación del problema:

Hay una persona que vive en un pequeño pueblo, un programador de alto rango que se encuentra paseando a su mascota como lo hace todos sus días de manera matutina, increíble y tristemente aparece de manera inesperada con la cara cubierta una persona la cual rapta a la mascota (sin chip ni identificación canica numérica), la agarra y se lleva en una sospechosa furgoneta, apenas la persona ingresó a la furgoneta se quitó la máscara por lo cual algunos testigos saben quién fue y otros que saben cómo llegar al culpable, necesitamos encontrar a la mascota de la manera más rápida posible.

Fase 2. Recopilación de la información:

Ya que tenemos el problema definido y conocemos las necesidades a cubrir, Bajo esta primicia se estructuró un plan de trabajo constante a lo largo de la búsqueda y la construcción de una solución efectiva, Dando

R1- Encontrar el camino más rápido para encontrar al perro.
R2- Buscar diferentes rutas.
R3- Mostrar ruta más corta.
R4- Ingresar el nombre de la persona a la cual le robaron la mascota.
R5- Generar un grafo (con mínimo 50 vértices).

Dentro de la información recopilada para hallar una solución encontramos las siguientes cosas:

Grafo:
Un grafo es una composición de un conjunto de objetos conocidos como nodos que se relacionan con otros nodos a través de un conjunto de conexiones conocidas como aristas. Un grafo en su totalidad es un par ordenado compuesto por vértices (v) y aristas (e); donde en la gran mayoría de los casos los vértices son de cuantificación finita.

Un grafo dirigido conocido también como dígrafo consta de un conjunto de vértices y aristas donde cada arista se asocia de forma unidireccional a través de una flecha con otro.

Los grafos no dirigidos son aquellos que constan de un conjunto de vértices que están conectados a un conjunto de aristas de forma no direccional.

Arista:
Son las líneas que unen los vértices de un grafo, existen diferentes tipos de aristas:
Aristas adyacentes: Dos aristas son adyacentes si convergen en el mismo vértice.

Aristas paralelas: Dos aristas son paralelas si los vértices iniciales y finales son el mismo vértice
Aristas cíclicas: Aristas que parten de un vértice para entrar en el mismo.
Una de las características de las aristas es el camino, que es el punto donde dos aristas se cruzan.

Vértice:
Los vértices constituyen uno de los dos elementos que forman un grafo.

Camino:
Se denomina camino a un conjunto de vértices interconectados por aristas. Dos vértices están conectados si hay un camino entre ellos.

Algoritmos de búsqueda BFS y DFS:
Una búsqueda en profundidad (DFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo. Su funcionamiento consiste en ir expandiendo cada uno de los nodos que va localizando, de forma recurrente (desde el nodo padre hacia el nodo hijo). Cuando ya no quedan más nodos que visitar en dicho camino, regresa al nodo predecesor, de modo que repite el mismo proceso con cada uno de los vecinos del nodo. Cabe resaltar que, si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.

Una búsqueda en anchura (BFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo), para luego explorar todos los vecinos de este nodo. A continuación, para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el grafo. Cabe resaltar que, si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.

Algoritmo de búsqueda Dijkstra:
Algoritmo de Dijkstra. También llamado algoritmo de caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

Como resumen tenemos :

Cliente	La policía
Usuario	Persona a la cual se le pierde el perro.

Requerimientos funcionales	• **R1**- Encontrar el camino más rápido para encontrar al perro. • **R2**- Buscar diferentes rutas. • **R3**- Mostrar ruta más corta. • **R4**- Ingresar el nombre de la persona a la cual le robaron la mascota. • **R5-** Generar un grafo (con mínimo 50 vértices).
Contexto del problema	Hay una persona que vive en un pequeño pueblo, un programador de alto rango que se encuentra paseando a su mascota como lo hace todos sus días de manera matutina, increíble y tristemente aparece de manera inesperada con la cara cubierta una persona la cual rapta a la mascota (sin chip ni identificación canica numérica), la agarra y se lleva en una sospechosa furgoneta, apenas la persona ingresó a la furgoneta se quitó la máscara por lo cual algunos testigos saben quién fue y otros que saben cómo llegar al culpable, necesitamos encontrar a la mascota de la manera más rápida posible.
Requerimientos no funcionales	• Que el programa sea hecho en java • Que el programa sea desarrollado en inglés.

Fase 3. Búsqueda de soluciones creativas:

Dentro de la búsqueda de soluciones creativas se puede destacar las siguientes técnicas de generación de ideas:

1. Tormenta de ideas: Fomenta la generación de ideas y soluciones a partir de un problema dado.

2. y si..? : Técnica que consiste en buscar métodos para solucionar problemas dados ciertos casos.

3. Mundos relacionados: A partir de este se observan diferentes enfoques del problema, por lo tanto es posible tener un panorama más amplio.

Dado estas técnicas el grupo decidió pensar desde diferentes puntos de vista, desde el que puede encontrar una solución hasta cómo actuaría una persona que en este contexto, le robaron a su perro, a continuación se mostrarán todas las ideas generadas.

Denunciar el secuestro por la internet:
Encontramos que una manera podría ser el interconectar a usuarios mediante la red, los cuales no tengan una relación directa persona a persona. Ayudando, a una mejora de la información bajo la comunicación.

Encontrados sitos en el diferentes foros web, que facilitan al entendimiento de la coneccion del Usuario con el mundo exterior, Por ello, se puede generalizar factores que se pueden ver alterados como por ejemplo: Las múltiples varias de tiempo involucrada en la búsqueda, las mejoras encontradas de la búsqueda, etc.

Poner volantes alrededor de la ciudad:
El colocar volantes tiene un gran factor efectivo dentro del ámbito urbano local y un poco a sus alrededores,Cabe resaltar lo sutil de esta idea, al tener una mejor percepción de la información y el acercamiento de la información dada por el volante , permite crear un vínculo el cual puede ayudar a encontrar si el lector del volante lo reconoce.
por último debemos entender que las mejoras que nos pueden dar el poner volantes es gigantesca al normalizar el poder entender los beneficios de darse a conocer , por ello, se debe de tener siempre encuenta una solucion facil, rapida y sencilla, aunque cabe aclarar que diferentes actividades pueden crear diferentes versiones a la hora de entregar información, esta se puede ver diversificada.

Colaborar con la policía en software:
Podemos recurrir a la autoridad competente que en este caso es la policía, esta autoridad nos puede ayudar a recuperar nuestra mascota ya que puede que se trate de un criminal que ya haya hecho esto muchas veces y se encuentra en búsqueda.

Teniendo en cuenta que afortunadamente es un pequeño pueblo, es fácil que la policía sepa quien se relaciona con quién, esta información la podemos usar para desarrollar un software el cual esté basado en un grafo y así por medio de las conexiones, hallemos la manera más rápida de saber quien es y cómo llegar a la persona comunicándose con sus relaciones.

Preguntar a conocidos:
Le preguntas a todas las personas que conoces que si conocen a alguien con las características del ladrón, para así dar con este de una manera u otra, por ende se necesita ubicar a las personas predestinadas, por último, se debe de entender el funcionamiento del habla a la hora de interactuar, bajo un lenguaje óptimo para comunicar una idea clara y concisa de esta, Al analizar la fase cuando se comunican se puede relacionar sus funcionalidades con conocer a

debidas personas o no, por ende, tenemos que tener en cuenta la tasa de persona conocidas que pueden llegar a compartir este mensaje con otras personas, Solo así se llegará a una

mayor amplitud de personas, y además la conexión unidireccional, ayuda a tener mayor cercanía con la persona.

Visitar Perreras:
Debemos conocer que quizás fue un error de la perrera local, Ya que últimamente se encuentra muchos animales sueltos y estos pueden realizar diferentes fechorías a lo largo y ancho de la ciudad, por último cabe resaltar de que la persona no se pudo ver si llevaba alguna identificación como agente civil, pero existen múltiples variables que pueden causar que estas no se hayan alcanzado a ver,etc. Aunque se necesita transicionar bajo la restricción de buscar en la zona donde el perro fue ultima vez visto.

Buscarlo por su propia cuenta:
En un momento de desesperación, lo primero que pasa por la mente de la persona es ir a buscarla por el pueblo por su cuenta, ir por todas las calles esperando que su mascota aparezca en algún momento, en cualquier lugar.

Fase 4. Transición de la formulación de ideas a los diseños preliminares:

Ideas descartadas:

Buscarlo por su propia cuenta:

Esta idea fue la primera en ser descartada ya que es la menos eficiente, se perdería mucho tiempo en tan solo buscar parte por parte al perro por cuenta de uno. Lo más óptimo para una persona que perdió su perro es acudir a la comunicación ya que es más fácil llegar a un perro preguntando o llegar mediante la comunicación con otras personas.

Preguntar a conocidos:

Cuando se trata de preguntarle a conocidos la opción que se debe tener en cuenta es poco viable, ya que teniendo en cuenta el contexto del problema se tardaría demasiado tiempo en encontrar al perro, y no solo eso, además de esto los conocidos de uno pueden no tener conexión alguna con el paradero del perro. En conclusión la idea está descartada por el hecho de que es poco viable y eficiente, puede que lleguemos a dar con el perro, sin embargo tardaremos mucho en llegar a encontrar una respuesta, lo cual convierte esto en una incertidumbre.

Visitar Perreras:

Visitar perreras es una manera viable, sin embargo una de las opciones más viables es recurrir a otro tipo de opciones que sean capaces de solucionar el problema de manera más rápida. Por lo tanto visitar perreras tomaría demasiado tiempo para ejecutar este proceso, además es muy alta la probabilidad de que el perro no esté en una perrera ya que dentro de este contexto es más fácil llegar al perro mediante otros medios.

Ideas no Descartada:

Denunciar el secuestro por la internet:

Como habíamos visto anteriormente una de las ideas más óptimas para dar con el paradero del perro es mediante la comunicación, si bien es algo muy fácil difundir por internet o redes sociales la desaparición de un perro. Esta idea es una idea a tener en cuenta para la solución del problema ya que como ya vimos es fácil dar con un perro, conectándonos con otra personas que probablemente tengan el conocimiento del paradero.

Colaborar con la policía en software:

La policía cuenta con un software especializado para dar con la búsqueda de alguien. Por lo tanto, teniendo en cuenta esto es una opción muy viable recurrir a la policía para dar con la búsqueda y paradero de un perro el cual está perdido. Un software permite realizar muchos procesos simultáneos, y dentro de estos procesos se puede concluir cuál es la manera más fácil de llegar a una respuesta.

Poner volantes alrededor de la ciudad:

Al igual que difundir la búsqueda del perro en internet, pegar volantes por la ciudad es una buena manera de encontrar a un perro, puesto que se puede saber usando la comunicación entre las personas el paradero del perro.

Fase 5. Evaluación y selección de la mejor solución:

Los criterios que se van a tener en cuenta para evaluar la mejor idea son los siguiente:

A. Eficiencia
B. Efectividad
C. Facilidad
D. Seguridad

Criterios opción A:
[3] Se soluciona el problema de manera rápida
[2] Se demora algo en solucionar el problema
[1] Se demora mucho en solucionar el problema

Criterios opción B:
[3] Da directamente con el ladrón
[2] Da con alguien relacionado al ladrón
[1] No da con el ladrón

Criterios opción C:
[3] Es mucho más fácil para la persona encontrar al perro
[2] La persona tarda en encontrar al perro
[1] La persona no logra encontrar al perro

Criterios opción D:
[3] Es más seguro para la persona encontrarlo de esa manera
[2] La persona toma ciertos riesgos de esa manera
[1] La persona toma altos riesgos de esa manera

Alternativa	Criterio A	Criterio B	Criterio C	Criterio D	Puntaje total
Denunciar secuestro por internet	2	2	3	3	10
Colaborar con la policía en software	3	3	3	3	12
Poner volantes alrededor de la ciudad	1	2	2	1	6

De las 3 ideas que no fueron descartadas podemos hacer una retroalimentación mediante la cual podemos encontrar la solución más óptima, sin embargo la manera más viable es usando el software de la policía.

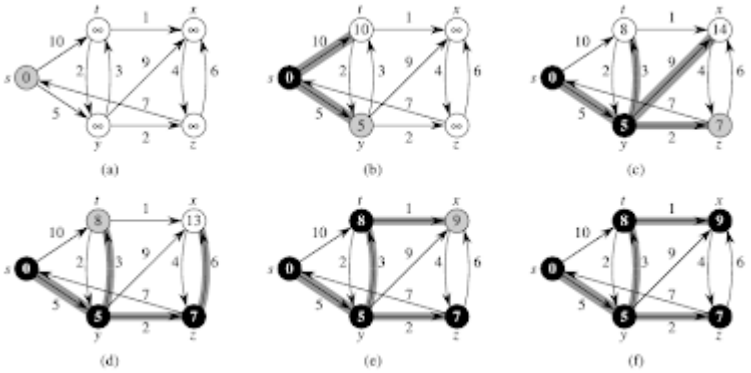
La manera más fácil de llegar a un perro es mediante la comunicación, que es una característica que la porta el hecho de poner volantes alrededor de la ciudad o también difundir la búsqueda del perro en internet. Estas dos maneras puede llegar a ser una sola

usando un software de búsqueda, software que tiene en este caso la policía, sabiendo que la policía cuenta con software especializado para la búsqueda, se puede concluir las siguientes cosas:

- El software de la policía use métodos de búsqueda como Dijkstra o BFS/DFS
- El software de búsqueda tiene como objetivo usar los métodos anteriormente mencionados los cuales permiten encontrar el camino más fácil de un punto a otro.

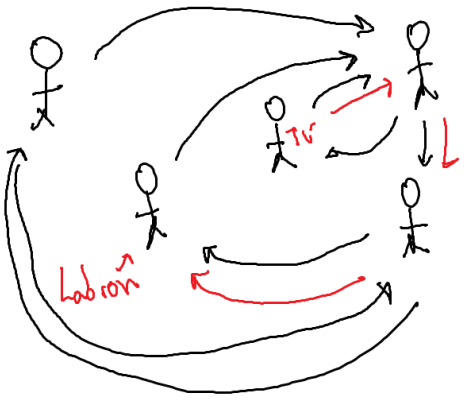
Analizando lo anteriormente visto, se puede ver como funciona el software de la policía:

Método de búsqueda usando el algoritmo de Dijkstra: Método que consiste en encontrar el camino más corto desde un vértice a otro, que en este caso es desde una persona hasta otra, si observamos de manera gráfica esto, podemos ver lo siguiente:



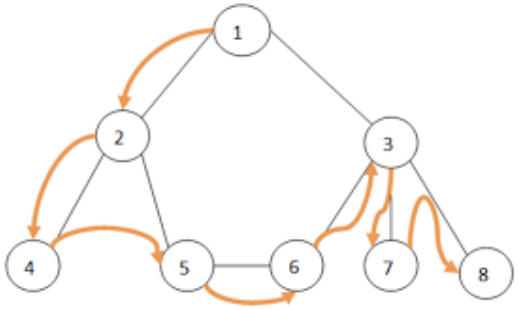
En esta observación podemos ver que el algoritmo de Dijkstra es capaz de hallar el camino más fácil desde un punto a otro.

Si pasamos el uso de este metodo de busqueda aplicado al problema podemos verlo de la siguiente manera:

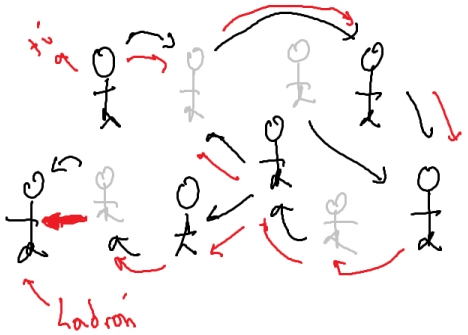


Analizando esta imagen podemos ver que la manera más fácil de llegar al ladrón del perro es mediante el camino más corto en el que se tiene en cuenta la conexión de las personas.

Método de búsqueda usando DFS: Método que recorre todo el grafo y sus conexiones, al recorrer persona (vértice) por persona, termina dando con el ladrón (vértice), podemos ver la representación de este método a continuación:



Observando esta imagen podemos ver que el algoritmo de dfs recorre el grafo desde un vértice hasta otro, que si lo ponemos en el contexto de este problema podemos ver esto como lo siguiente:



Lo que hacemos en el método de dfs es preguntar persona por persona, así no tengan conexión alguna con el paradero del perro. Lo que en consecuencia al preguntar a cada persona se va a dar con el paradero del perro.

Paso 6. Preparación de Informes y Especificaciones:

Especificación del problema:

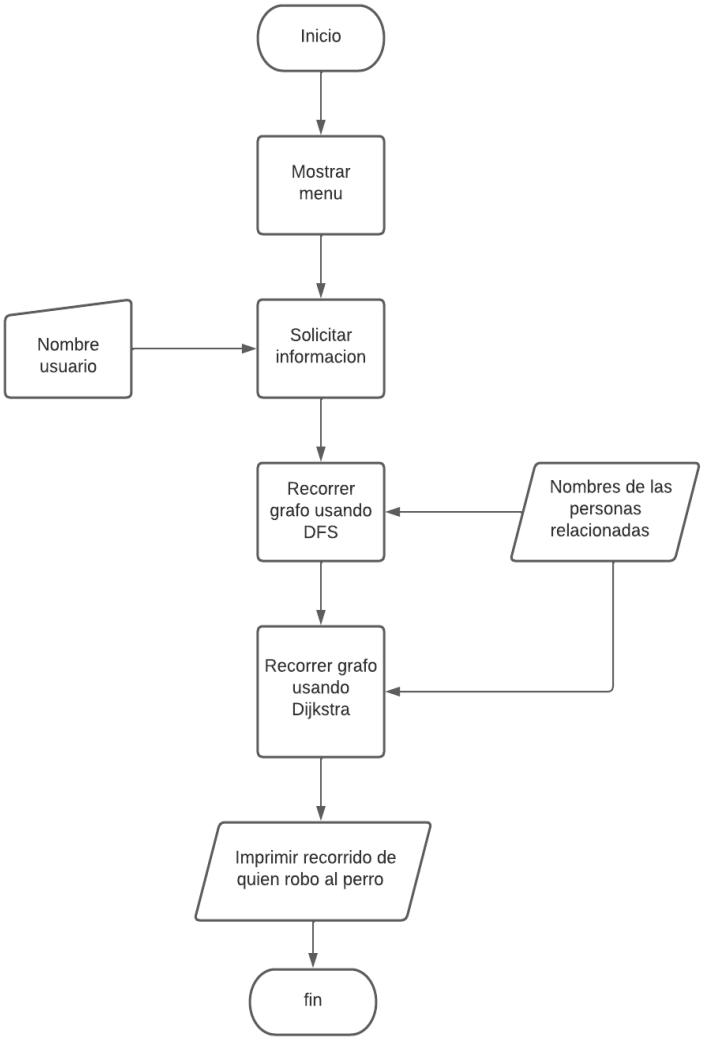
Problema: Hallar la ubicación de un perro y la persona que la robo.

Entradas: Nombre de la persona

Salida: Nombres de las personas por las cuales se tuvo que hacer el recorrido para llegar hasta el perro.

Consideraciones: Conexiones de las personas entre ellas, esto en contexto de grafos es llamado como vértices y cada vértice tiene un peso.

Diagrama de flujo:



Paso 7: Implementación del diseño:

El código se implementa en java, y se usa javaFx para la interfaz de usuario.

Lista de tareas a implementar:

1. *Mostrar menú en pantalla*
2. *Añadir vértice*
3. *Realizar recorrido Dijkstra*
4. *Comparar vértices adyacentes en aristas*
5. *Imprimir recorrido*
6. *Comparar vértices*
7. *Añadir arista*

TADS:

Graph
Invariante: Contiene vértices y aristas.
Estructura de Datos: **pq:PriorityQueue<Vertex>** **vertexes: ArrayList<Vertex>**
OPERACIONES: **Graph(): void** **newVertex(name: Sting) : String** **addEdge(source : String, dest : String, weight : int) : void** **search(goal : String) : Vertex** **BFS(name : String) : void** **dijkstra(init : String, finish : String) : void** **compAdjEdges(s : Vertex, w : int) : void** **printPath(c : Vertex) : void**

Añadir vértice:

Nombre	newVertex(String name) → String
Descripción	Método que añade un vértice al arrayList de vértices
Pre	El arraylist está instanciado
Pos	devuelve el nombre del vértice agregado

Añadir Arista:

Nombre	addEdge(String name, String dest, int weight) → void
Descripción	añade una arista entre dos vértices
Pre	los vértices existen
Pos	se crea una arista entre los dos vértices

Realizar recorrido Dijkstra:

Nombre	BFS(String name) → void
Descripción	Recorre el grafo en anchura
Pre	Existe un grafo
Pos	Todos los vértices tienen un color de 2 o 3

Comparar vértices en las aristas:

Nombre	dijkstra(String init, String finish) → void
Descripción	Recorre el grafo para elegir el camino más corto entre el inicio y final
Pre	El grafo existe. el vértice de inicio y final existe
Pos	Se elige la ruta más corta

Imprimir recorrido:

Nombre	compAdjEdges(Vertex s, int w) → void
Descripción	Recorre los vértices adyacentes del vértice de inicio, para tomar su distancia y comparar la mínima. Esta distancia se suma para determinar la más corta
Pre	Existe el vértice
Pos	Se suma la distancia del adyacente con menor distancia.

Comparar vértices:

Nombre	printPath(Vertex c) → void
Descripción	Concatena los nombres de los vértices por los vértices que conforman el camino más corto al vértice deseado
Pre	Existe el vértice
Pos	el mensaje con el camino más corto

Diseño de pruebas:

Bajo esta idea de diseñar casos de pruebas eficientes en nuestro algoritmo, se nos ocurrió la idea de realizar las siguientes, con el fin de cubrir todas las funcionalidades de nuestro algoritmo, Teniendo en cuenta el criterio de las tablas de taguchi (L4) , nos damos cuenta que podemos crear la siguiente versión de Casos de pruebas.

Tipo de Grafo	Cantidad de Vértices	Cantidad de Aristas	Nodo fuente	Algoritmos a Usar
No conexo	5	4	No alejado	BSF, Dijkstra, newVertex,addEdge,search,compAdjEdges.
Conexo		>6	Alejado	
No conexo		>6	alejado	
Conexo		4	No alejado	

Tras finalizar la primera etapa, nos encontramos que solo 2 casos de pruebas son suficientes para Cerrar nuestros test, ya que estos cubrirán todos los Campos que su segundo caso de prueba tiene, por ende terminamos con las siguientes:

Tipo de Grafo	Cantidad de Vértices	Cantidad de Aristas	Nodo fuente	Algoritmos a Usar
No conexo	5	4	No alejado	BSF, Dijkstra, newVertex,addEdge,search,compAdjEdges.
Conexo		>6	Alejado	

Al finalizar nuestros test esperamos un resultado esperado de más del 80 % esto, para asumir que nuestro programa tiene todas las pruebas necesarias, desde las unitarias, hasta las pruebas de integración que se pueden ver al Correr nuestro código, junto esto podríamos realizar un prueba con técnica de caja negra que nos describira el resultado esperado de cada Test si se necesita más información acerca de las diferentes funcionalidades establecidas a lo largo del desarrollo de nuestro programa, Cabe aclarar que faltaría verificar los requerimientos no funcionales, que se pueden ver afectado a lo largo del desarrollo.

Usando librerías como Junit y Júpiter, vamos a tener los resultados finales esperados para cada Caso de Prueba.

Referencias:

https://www.techiedelight.com/es/depth-first-search-dfs-vs-breadth-first-search-bfs/

https://www.codingame.com/playgrounds/7656/los-caminos-mas-cortos-con-el-algoritmo-de-dijkstra/el-algoritmo-de-dijkstra

https://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos

https://www.partesdel.com/partes_del_grafo.html

https://www.emprendeaconciencia.com/blog/tecnicas-y-herramientas-de-ideas