Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

MIGUEL RAMSAUER NETO

Um Objeto de Aprendizagem para o ensino de Classificação

Assis
2014

Av. Getúlio Vargas, 1200 – Vila Nova Santana – Assis – SP – 19807-634
Fone/Fax: (0XX18) 3302 1050 homepage: www.fema.edu.br

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

Um Objeto de Aprendizagem para o ensino de Classificação

Projeto apresentado ao curso de Analise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, com objetivo de estudar algoritmos de classificação e desenvolver um Objeto de Aprendizagem para auxiliar no ensino desse conteúdo.

Orientando: Miguel Ramsauer Neto
Orientador: Doutor Luiz Ricardo Begosso

Área de Pesquisa: Ciências Exatas e da Terra

Assis
2014

FICHA CATALOGRÁFICA

Ramsauer Neto, Miguel
Um Objeto de Aprendizagem para o ensino de Classificação / Miguel Ramsauer Neto. Fundação Educacional do Município de Assis – FEMA – Assis, 2014.
26 Páginas.

Orientador: Doutor Luiz Ricardo Begosso
Programa de Iniciação Cientifica – Instituto Municipal de Ensino Superior de Assis – IMESA.
1. Algoritmos de Classificação. 2. Objeto de Aprendizagem.

CDD: 001.61
Biblioteca da FEMA

Resumo

Este trabalho descreve a pesquisa sobre objetos de aprendizagem e classificação de dados assim como a produção de um objeto de aprendizagem para auxiliar no ensino de classificação.
Durante a pesquisa foi notado o grande potencial que objetos de aprendizagem tem de auxiliar no ensino dos mais variados tópicos.
No decorrer do trabalho ficou claro a importância dos métodos de classificação, as distinções entre eles bem como a escolha do método adequado pode influenciar aplicações de maneira substancial.
Na criação do objeto de aprendizagem foram utilizadas as ferramentas HTML5, CSS e Javascript para a criação de um web site com objetivo de auxiliar no ensino de classificação de dados.

Lista de ilustrações

Sumário

1. Introdução

A tecnologia da informação e da comunicação tem contribuído com diversas áreas do conhecimento, em especial, destaca-se a área da educação. No tocante ao ensino de conceitos básicos da computação, objeto desse trabalho, a literatura apresenta alguns esforços que objetivam auxiliar no processo de ensino aprendizagem: o *Scratch* é um ambiente que proporciona a construção de programas e foi criado com o objetivo de facilitar a aprendizagem de pessoas iniciantes em programação. Ele foi desenvolvido pelo MIT (Massachusetts Institute of Technology) com a linguagem de programação *Squeak* (Ford, 2009); o *Greenfoot*, desenvolvido pela Universidade de Kent, na Inglaterra, e pela Universidade de Deakin, na Austrália tem a finalidade de ensinar programação orientada a objetos a partir da construção de cenários utilizando o ambiente preparado para o desenvolvimento de jogos. O *Greenfoot* pressupõe que o estudante possua conhecimento prévio de conceitos básicos de programação (Kölling e Henriksen, 2005); e, finalmente, o ALICE é um ambiente de programação 3D, desenvolvido especialmente para alunos que terão sua primeira experiência com programação orientada a objetos. O software permite que o aluno aprenda conceitos fundamentais de programação criando animações e jogos. No ALICE, objetos 3D como pessoas, animais, veículos, etc. formam um mundo virtual onde os alunos criam programas para animar tais objetos, (ALICE, 2013).

Experiências relatadas com o uso desses softwares apontam para o fato de que, apesar do sucesso de tais experiências, muito ainda precisa ser feito nessa direção.

Uma alternativa que tem contribuído com o processo ensino-aprendizagem é a utilização de Objetos de Aprendizagem (OA). Estes recursos tendem a facilitar e proporcionar suporte tecnológico aos processos educacionais.

De acordo com IEEE (2013), OA são entidades digitais ou não-digitais que podem ser usadas e reutilizadas durante um processo de aprendizagem sustentado por tecnologia.

Os Objetos de Aprendizagem podem ser utilizados de forma discreta, abordando uma parte de conteúdo que será trabalhado com os alunos. O docente, em vez de fornecer todo o material do curso por meio de apostilas, livros, artigos ou anotações de aula, utiliza

os OA que, por sua vez, contribuem no sentido de mediar e qualificar o processo de ensino-aprendizagem.

Linux Educacional (2013), destaca que aplicações dos Objetos de Aprendizagem podem ser: animações, vídeos, simulações, páginas HTML, imagens, etc.

Para Learning Objects (2013), os OA apresentam as seguintes características:

- Eles são autocontidos: cada OA pode ser utilizado de forma independente;
- Eles são reutilizáveis: um OA pode ser utilizado para contextos diferentes e para múltiplos propósitos;
- Eles podem ser agregados: OA podem ser agrupados em coleções maiores, permitindo o uso dentro de uma estrutura tradicional de curso;
- Eles podem ser indexados e armazenados em repositórios: os OA devem possuir informação que o descreva, permitindo ser facilmente encontrado numa busca. Essa característica promove o uso do OA por outras pessoas.

2. Objetos de Aprendizagem

Os avanços tecnológicos nas últimas décadas mudaram para sempre muitos aspectos da sociedade como um todo. Apesar das instituições de ensino terem mantido basicamente o mesmo modelo de ensino, com o passar dos anos também foram criadas e propostas inúmeras ideias de ensino alternativas com os mais variados processos e resultados.

A internet modificou para sempre a forma de comunicação entre as pessoas, e foi pensando nessa incrível tecnologia que os objetos de aprendizagem foram propostos, para ensinar qualquer assunto, para qualquer um que tiver acesso a um computador em condições de suportar o objeto de aprendizagem.

Objetos de aprendizagem se destacam de outros métodos de ensino pelo seu potencial para a reusabilidade, generatividade, adaptabilidade e escalabilidade (Hodgins, 2000; Urdan & Weggen, 2000; Gibbons, Nelson, & Richards, 2000).
Eles tem como base o conceito de programação orientada a objetos, que consiste na criação de "objetos" ou módulos que podem ser reutilizados em situações diferentes e para fins variados, e essa é a ideia principal para todo o conceito.
O intuito por trás dessa ideia é de que uma vez que um objeto de aprendizagem é concebido, ele pode ser reutilizado para ensinar em contextos diferentes, além de que por serem distribuídos através da internet eles podem ser usados por um número incontável de pessoas e a da facilidade de serem distribuídos.
O nome "Objeto de aprendizagem" foi possivelmente tirado do trabalho de Wayne Hodgins que em seu grupo de trabalho no CedMA chamado "Arquiteturas de aprendizagem, API's e Objetos de Aprendizagem".
Os objetos de aprendizagem se destacam também por serem geralmente bem menores que um curso inteiro por exemplo, podem ser combinados com outros objetos ao critério do usuário para um fim específico, criando assim uma combinação de objetos de aprendizagem que atende a necessidade do usuário, dando ainda mais liberdade para adicionar ou remover componentes a critério de quem está ensinando ou aprendendo.

Qualquer recurso digital que pode ser usado para apoiar o aprendizado pode ser considerado um objeto de aprendizagem, sendo assim até mesmo fotos e imagens, vídeo ou áudio ao vivo ou gravado, até mesmo aplicações disponíveis na internet como uma calculadora podem entrar na categoria de objetos de aprendizagem.

Assim como páginas na *web*, que combinam textos e múltiplas formas de mídia são objetos de aprendizagem mais complexos que demonstram o fator da reusabilidade dos OAs, uma página na web que ensina sobre as placas tectônicas e sua relação com terremotos, por exemplo, é possível retirar a parte que ensina sobre as placas tectônicas e utilizadas em outro objeto de aprendizagem sobre a formação da crosta terrestre, assim reutilizando apenas o conteúdo necessário para o objetivo de quem ensina.

3. Classificação de Dados

Classificação de dados ou ordenação de dados é uma técnica muito usada em todas as áreas da computação, ela é essencial para a optimização das tarefas, com os dados classificados tarefas específicas ficam muito mais rápidas e eficientes, em seu livro *"Information Anxiety"*, Richard Saul Wurman propõe que os principais criterios para a classificação são: localização, alfabético, tempo, categoria e hierarquia (LATCH).
Por exemplo: Toda vez que uma busca é feita no Google o retorno da pesquisa é classificado por hierarquia, baseado em seu sistema interno de pontuação para cada site.
Na maioria das vezes a ordem ascendente é a usada como padrão na classificação e.g. A a Z ou 0 a 9.

Classificar os dados serve diversos propósitos, mas geralmente tem em comum organizá-los de acordo com um critério e facilitar o acesso, pesquisa e entendimento dos dados classificados, desde listas de produtos a arquivos em um computador.

Na computação existem inúmeros métodos de classificação, neste trabalho serão discutidos e exemplificados os seguintes métodos: bolha, inserção, seleção, contagem, contagem por distribuição, *shell* e *quicksort*.

Serão também usados vetores como estrutura de dados e pseudocódigo para demonstrar a teoria e o funcionamento dos mesmos.

3.1. Método Bolha (*bubble sort*).

Este método recebe esse nome porque os maiores valores vão se agrupando no final do vetor, imitando como bolhas flutuam até a superfície da água.
Esse método é um dos mais simples e por sua vez um dos menos eficazes quando o número de elementos a serem classificados é grande.
A ideia deste método é encontrar o elemento com o maior valor e reposicioná-lo na última posição do vetor, até que todos estejam em ordem.

Construção e pseudocódigo:
Variáveis:
'i' e 'j' são variáveis que fazem o controle do vetor para que ele faça todas as passagens necessárias.
'temp' é a variável que guarda o valor de um dos dados que será trocado.
'n' é o tamanho do vetor a ser classificado que ficara constante durante toda a execução, ou seja, o tamanho do vetor é sempre o mesmo.

'vetor[]' é o vetor que ira ser classificado, neste caso suponha que o vetor já tenha todos os dados.

Código:

```
i=0
while(i < n) {

      j=0
      while (j < n) {
        if (vetor[j] > vetor[j+1]) {
            temp = vetor [j]
            vetor[j] = vetor[j+1]
            vetor[j+1] = temp
            }
    j++
  }
i++
}
```

Entendendo o código acima:
Para cada novo valor da variável 'i' o valor de 'j' retorna a zero, para que ele faça as comparações entre os valores novamente e se não estiverem em ordem continuar a classificá-los.
O "*core*" ou núcleo do programa é a comparação entre os valores dentro do vetor, a repetição dessa comparação e a troca simples se ela for verdadeira é o mecanismo que classifica os valores.

```
 if (vetor[j] > vetor[j+1]) {      // Se o valor de uma dada posição for maior que o da próxima
            temp = vetor [j]       // A variável 'temp' recebe o maior valor
            vetor[j] = vetor[j+1] // O endereço onde estava o maior valor recebe o menor
            vetor[j+1] = temp     // O endereço do menor recebe o conteúdo de 'temp'
            }
```

Assim toda vez que um valor maior estiver posicionado antes de um menor eles trocarão de lugar.

Mas esse código do jeito que está apresenta um pequeno problema, não importa se o vetor em questão esteja em ordem ou não ele continuara a fazer comparações.
Para resolver esse problema se usa uma *flag*, bandeira em inglês, para sinalizar ao algoritmo que os valores estão desordenados e precisam ser classificados.
Para esse novo código aprimorado apenas acrescenta-se uma nova variável do tipo inteiro, ela vai se chamar '*flag*'.

Novo código:

```
i=0
flag=0
while(i < n && flag = 0) {
       flag=1
       j=0
       while (j < n) {
         if (vetor[j] > vetor[j+1]) {
             temp = vetor [j]
             vetor[j] = vetor[j+1]
             vetor[j+1] = temp
             flag = 0
             }
     j++
  }
i++
}
```

Como pode se observar o algoritmo continua o mesmo, a diferença é que agora temos uma nova condição de parada.
A variável 'flag' começa com o valor 0 indicando que o vetor está fora de ordem, ao entrar na repetição ela recebe o valor 1, apontando que o vetor foi ordenado, se em nenhuma das vezes que a comparação for feita o vetor tiver que trocar as posições de dois valores isso significa que o vetor está ordenado corretamente e 'flag' continuara valendo 1.
Na próxima vez que a condição do "*while*" for verificada ela será falsa e o algoritmo se encerrara uma vez que a classificação foi terminada.

No entanto, se a comparação "if (vetor[j] > vetor[j+1])" for verdadeira o valor de 'flag' é 0 isso mostra para o algoritmo que o vetor não estava ordenado e faz com que ele seja executado no mínimo mais uma vez.

3.2. Método de Inserção

Esse método é simples e funciona relativamente bem quanto a quantidades pequenas de dados e consiste em percorrer o vetor da esquerda para a direita e organizar os valores a esquerda, inserindo o valor na sua posição final, comparando-o com os anteriores e trocando enquanto ele for menor.

Construção e pseudocódigo:
A ideia por trás desse algoritmo é a separar o vetor em duas partes, uma a esquerda já ordenada e outra a direita ainda a ordenar, o código a seguir mostra como isso é implementado.

```
Void Inserção(int Vetor[], int tamanho)
{
 int j, i, aux;
while (i < tamanho)
  {
        j = tamanho;
        aux = Vetor[i];
        while(temp < A[j-1])
        {
        A[j] = A[j-1];       // Deslocamento – abre espaço para o elemento a ser inserido //
        j--;
        }
        A[j] = temp;          // Coloca o elemento na posição correta //
  }
}
```

3.3. Método de Seleção

A ideia por trás deste método de classificação é simples, localizar o menor valor e posicionar ele na primeira posição, o segundo menor e colocar ele na segunda posição e assim por diante até que o vetor inteiro esteja ordenado.

Como os algoritmos citados anteriormente neste trabalho ele também é relativamente simples, veja a seguir o pseudocódigo:

```
void selection_sort(int num[], int tam) //a função recebe o vetor e o tamanho
{
 int i, j, min, aux;
 for (i = 0; i < (tam-1); i++)
   {
    min = i;
    for (j = (i+1); j < tam; j++) { //esse loop encontra o menor elemento
      if(num[j] < num[min]) {
        min = j;
      }
    }
    if (i!= min) {
      aux = num[i];
      num[i] = num[min];
      num[min] = aux;
    }
  }
}
```

3.4. Método de Contagem

No método de ordenação por contagem é importante notar que se usam 3 vetores em vez de 1 ou dois como nos métodos anteriores, isso se deve ao fato de que existe um vetor com os dados originais, um para ser feita a contagem e outro para ser armazenado o resultado.

```
Void contagem(int vet[],int tam){
int i, j, a=0;
int cont[10];
   for(int i=0; i< 10; i++){ //preenche o vetor de contagem com 0
      cont[i] = 0;}
   for(i = 0; i < 10; i++){ //adiciona 1 no valor do ''cont'' sempre que tiver um valor maior
      for(j = i+1; j < 10; j++){
         if(vet[i] > vet[j]){
            (cont[i])++;
         }
         else{
            (cont[j])++;
         }
      }
   }
   for(i=0;i<10;i++){ //preenche o vetor de saida baseado no vetor de contagem
      saida[cont[i]]= vet[i];
   }
```

Ele funciona pelo seguinte, o vetor de contagem tem em seu valor a posição do vetor original onde ele ficará no vetor ordenado, o grande diferencial desse método é o fato de que ele não utiliza de trocas entre valores como o método bolha por exemplo.

3.5. Método Shell

O método de Shell é baseado no método de inserção, e utiliza um conceito chamado de ''passos largos'', que basicamente significa que as comparações feitas por ele comparam elementos do vetor distantes um do outro e por causa disso ele é bastante eficiente para quantidades de dados de tamanho médio (por volta de 5.000 registros).

Para implementar a ideia de passos largos são usados incrementos, que nada mais são do que o tamanho desse passo largo, nota-se que o incremento tem seu valor diminuindo ao final de cada passagem até que seu valor se torne 1 e seja feita a última passagem e certifica-se de que o vetor está em ordem.

Não existem regras para o valor do incremento e nem de como ele vai ser diminuindo, apenas que o valor do incremento seja maior que 1 e no final ele tenha o valor igual a 1.

O código a seguir é uma das interpretações dessa técnica:

```
void shell(int vet[],int in){
 int inc[] = {in, in/2, in/4, in/8}; //os incrementos vão ser armazenados em um vetor
    int i=0, a=0, aux;
    int fim = 10;
    while(a < 4){
       while(i + inc[a] < fim){
          int j = i;
             while(vet[inc[a] + j] <= vet[j]){// aqui é basicamente o método de inserção
                aux = vet[j];
                vet[j] = vet[j + inc[a]];
                vet[j + inc[a]] = aux;
                j--;
             }
         i++;
      }
      a++;
      i=0;
   }
}
```

3.6. Método Quicksort

O método de classificação Quicksort é o mais rápido dentre os apresentados nesta pesquisa, ele pode classificar enormes quantidades de dados em pouco tempo, mas devido a sua natureza mais complexa, como por exemplo a utilização da recursividade para sua execução e necessita de mais poder de processamento do que os outros, então não é recomendado para utilização em quantidades de dados menores uma vez que seu custo com relação a máquina e o código é maior do que os benefícios nesses casos.

O funcionamento dele tem como base o conceito de um pivô, que divide o vetor em dois subvetores, porém como ele é uma função recursiva os subvetores também são divididos em dois e assim por diante até que tenha apenas 2 elementos na última execução e então ele classifica todos os elementos a partir dessa condição.

```
void quickSort(int vetor[], int primeiro, int último) {
   int i = primeiro, j = ultimo, pivo = vetor[(j+i)/2], aux;
   while(i <= j) //utiliza o pivo para dividir o vetor
   {
      while(vetor[i] < pivo && i < direita)
      {
         i++;
      }
      while(vetor[j] > pivo && j > esquerda)
      {
         j--;
      }
      if(i <= j)//troca simples
      {
         aux = vetor[i];
         vetor[i] = vetor[j];
         vetor[j] = aux;
         i++; j--;
      }
   }
   if(j > esquerda)//recursividade
   {
      quickSort(vetor, esquerda, j);
   }
   if(i < direita)//recursividade
   {
      quickSort(vetor, i, direita);
   }
}
```

A função recebe o vetor a ser ordenado, o primeiro valor que geralmente é 0 e o ultimo valor que é o tamanho do vetor, isso na primeira passagem já nas demais o ultimo ou primeiro serão o pivô.

Como pode ser observado no código anterior é um algoritmo que utiliza trocas entre os valores, seguindo a lógica dos métodos como bolha e seleção porém implementados de uma maneira muito mais eficiente.

Lembrando sempre que a eficiência se vale no quesito de um número elevado de dados, utilizar Quicksort ao invés da bolha, inserção ou seleção para um organizar um vetor de poucos elementos é ineficiente pois exige muito mais da máquina do que o necessário, é sempre importante levar em consideração a optimização de um programa, da mesma maneira que utilizar os métodos citados anteriormente em grandes quantidades de dados em vez de Quicksort eleva e muito o tempo de execução da aplicação.

4. Objeto de Aprendizagem Para o Ensino de Classificação

O objeto de aprendizagem desenvolvido no trabalho se trata de um web site onde os usuários poderão visualizar o funcionamento dos diversos algoritmos de classificação através de animações interativas criadas utilizando o recurso *canvas* da tecnologia HTML5 e Javascript, explicações teóricas simples e exemplos de códigos.

A página oferece uma interface de fácil navegação para a escolha do tópico que o usuário deseja.

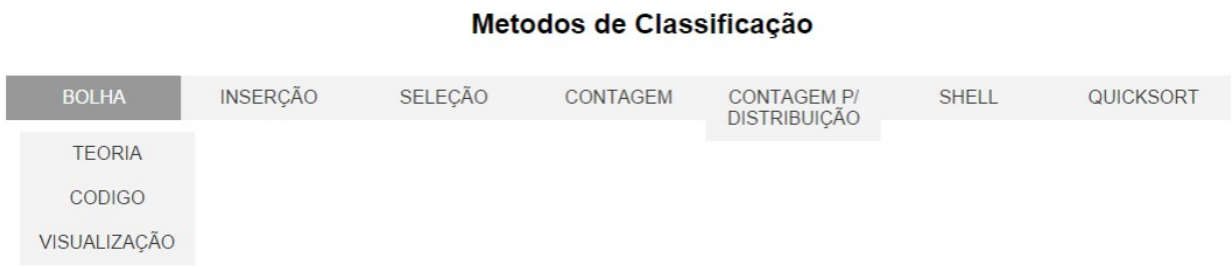

Figura 1. Interface de Navegação da página.

O objeto de aprendizagem conta com uma página com uma breve explicação teórica do algoritmo em questão.

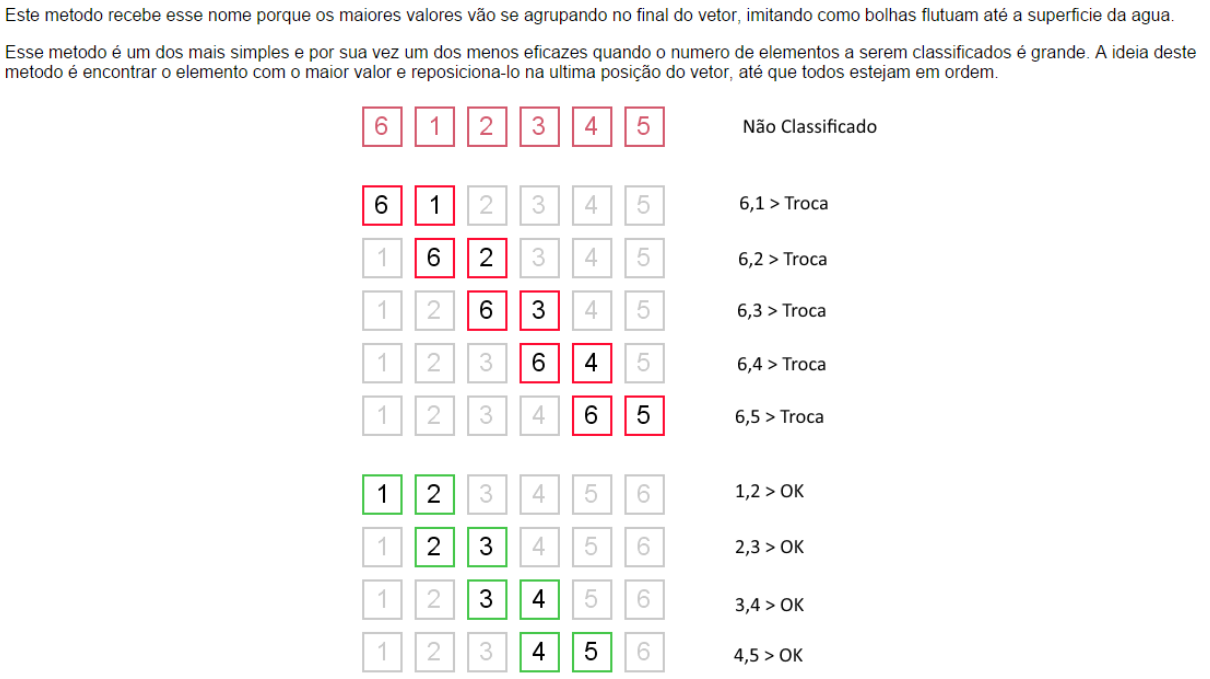

Figura 2. Teoria.

Para complementar a teoria a página também oferece aos usuários a possibilidade de ver pseudocódigos dos algoritmos, como observados na figura a seguir:

```
Novo codigo:
i=0
flag=0
while( i < n && flag = 0) {
flag=1
j=0
while ( j < n ) {
if ( vetor[j] > vetor[j+1] ) {
temp = vetor [j]
vetor[j] = vetor[j+1]
vetor[j+1] = temp
flag = 0
}
j++
}
i++
}
```

Figura 3. Pseudocódigo.

Para complementar as explicações teóricas e os códigos mostrados também são oferecidas visualizações do funcionamento de cada algoritmo, que deixa fácil o entendimento das diferenças em performance entre os algoritmos estudados e do comportamento de cada um.

Levando assim a um melhor entendimento de como cada método de classificação funciona e como os códigos aprendidos em aula se aplicam em uma situação real.

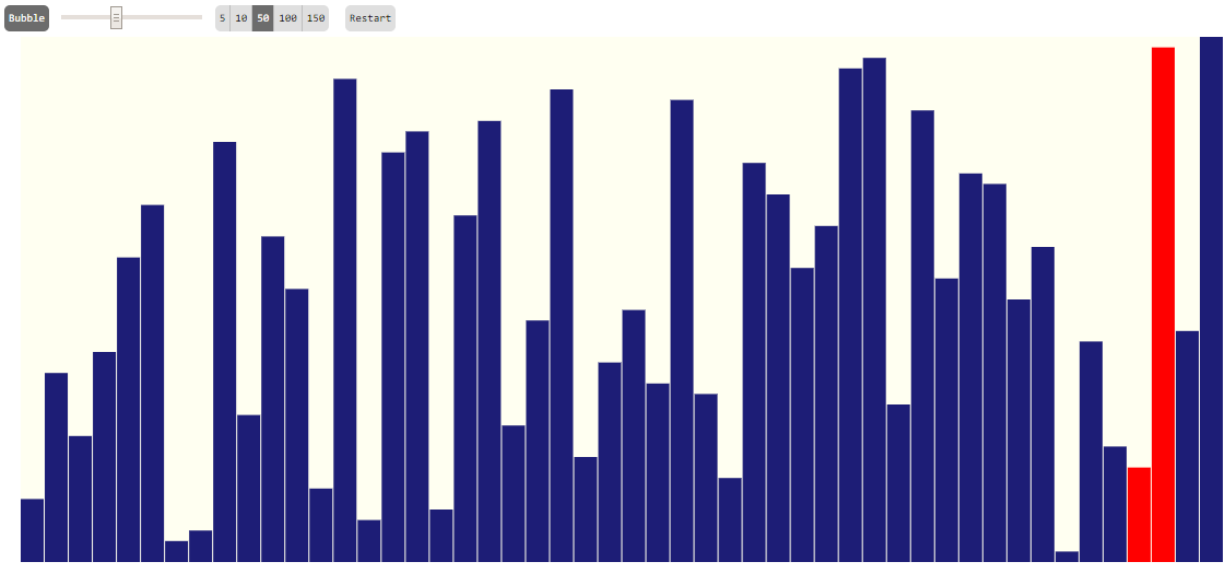

Figura 4. Visualização de um dos algoritmos.

Conclusão

Em relação aos objetos de aprendizagem foi concluido que eles são uma ferramenta fantástica para o ensino dos mais diversos temas e assuntos, porém não vem sendo usados para auxiliar no ensino tanto quanto poderiam.

A ideia de que os alunos da instituição possam produzir tais ferramentas e pesquisar sobre um determinado tema relacionado ao curso é muito interessante devido ao fato de que em curto prazo a FEMA possa contar com uma biblioteca de objetos de aprendizagem que auxiliem aos alunos em seu ensino.

No decorrer da implementação do objeto de aprendizagem foram estudados os recursos da tecnologia HTML5 e Javascript, que são relativamente novas e capazes de produzir resultados estonteantes para desenvolvimento web e de qualidade e complexidade formidáveis.

Em relação a classificação de dados foi notado que desde os mais simples processos até operações complexas é crucial a organização e ordenação de dados, arquivos e registros.

Muito do que é usado no dia a dia tem relação com processos de classificação sem o usuário final se dar conta, como buscas em um site como o Google, onde ocorre um processo complexo de classificação dos resultados o torna tão útil e indispensável na vida moderna.

Os algoritmos estudados para está pesquisa variam de simples trocas entre valores maiores e menores se repetindo incessantemente até que estejam em ordem até algoritmos que utilizam funções recursivas complexas que tem capacidade de organizar milhares de dados com eficiência.

REFERENCIAS

Gosling, James., Harrison, Jason. Sorting Algorithms. University of British Columbia, Vancouver, Canada. Disponível em <http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html> Acesso em 28 Nov. 2014.

Kruse Robert L., Ryba, Alex. Data Structures and Program Design in C++. Disponível em <http://iclass.iuea.ac.ug/intranet/E-books/INFORMATION%20TECHNOLOGY/C++%20and%20C%20programming/Data%20Structures%20and%20Program%20Design%20in%20C++%20(Robert%20Kruse,%20Alexander%20Ryba%3B%20Prentice%20Hall).pdf> Acesso em 28 Nov. 2014.

Lúcia Prata, Carmem., Aun de Azevedo Nascimento, Anna Christina. Objetos de aprendizagem: uma proposta de recurso pedagógico/Organização 2007.

Norahiko. ソートアルゴリズムを映像化してみた. Disponível em <http://jsdo.it/norahiko/oxly> Acesso em 28 Nov. 2014.

R. Martin, David. Sorting Algorithms Animations. Disponível em <http://www.sorting-algorithms.com> Acesso em 28 Nov. 2014.

S. Adamchik, Victor. Sorting. CMU. Disponível em <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Sorting%20Algorithms/sorting.html> Acesso em 28 Nov. 2014.

Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy.