

4^a SESIÓN

ALGORÍTMICA

BACKTRACKING

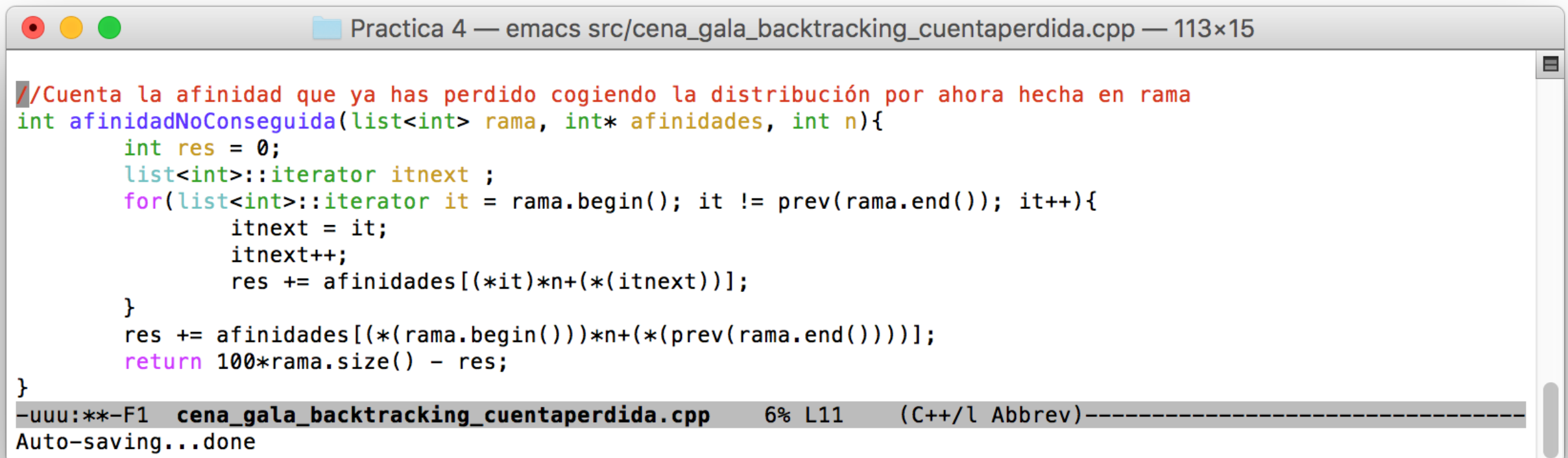
CENA DE GALA BACKTRACKING CUENTA PERDIDA – (ALGORITMO FACTORIAL)

```
Practica 4 — emacs src/cena_gala_backtracking_cuentaperdida.cpp — 146x32

list<int> backtracking(list<int> rama, int* afinidades, list<int> sinuser, int &bestafinlost, int n){
    if(afinidadNoConseguida(rama,afinidades,n) >= bestafinlost){ // Si la afinidad perdida es más grande que la afinidad
        list<int> res; // conseguida en la mejor distribucion hasta el momento, desiste .
        res.clear();
        return res;
    }else{
        int tam = sinuser.size();
        if(tam <= 1){ // Si queda una persona por sentar, men, has llegado
            rama.splice(rama.end(),sinuser);
            bestafinlost = afinidadNoConseguida(rama,afinidades,n);
            return rama;
        }else{
            list<int>::iterator it;
            list<int> res;
            list<int> aux2;
            for(it = sinuser.begin(); it != sinuser.end(); it++){ // Crea una rama por cada persona que no este sentada
                int aux = *it;
                it = sinuser.erase(it);
                rama.push_back(aux);
                aux2 = backtracking(rama, afinidades, sinuser, bestafinlost, n);
                if(aux2.size() != 0){
                    res = aux2;
                }
                it = sinuser.emplace(it,aux);
                rama.remove(aux);
            }
            return res;
        }
    }
}

-uuu:---F1 cena_gala_backtracking_cuentaperdida.cpp 20% L31 (C++/l Abbrev)-----
```

CENA DE GALA BACKTRACKING CUENTA PERDIDA – (ALGORITMO FACTORIAL)



The image shows a screenshot of an Emacs editor window. The title bar at the top reads "Practica 4 — emacs src/cena_gala_backtracking_cuentaperdida.cpp — 113x15". The code is written in C++ and is color-coded. It defines a function `afinidadNoConseguida` that takes a `list<int>` named `rama`, an `int*` named `afinidades`, and an `int` named `n`. The function calculates a result `res` by iterating through the `rama` list and comparing elements with the `afinidades` array. The status bar at the bottom shows "-uuu:**-F1 cena_gala_backtracking_cuentaperdida.cpp 6% L11 (C++/l Abbrev)-----" and "Auto-saving...done".

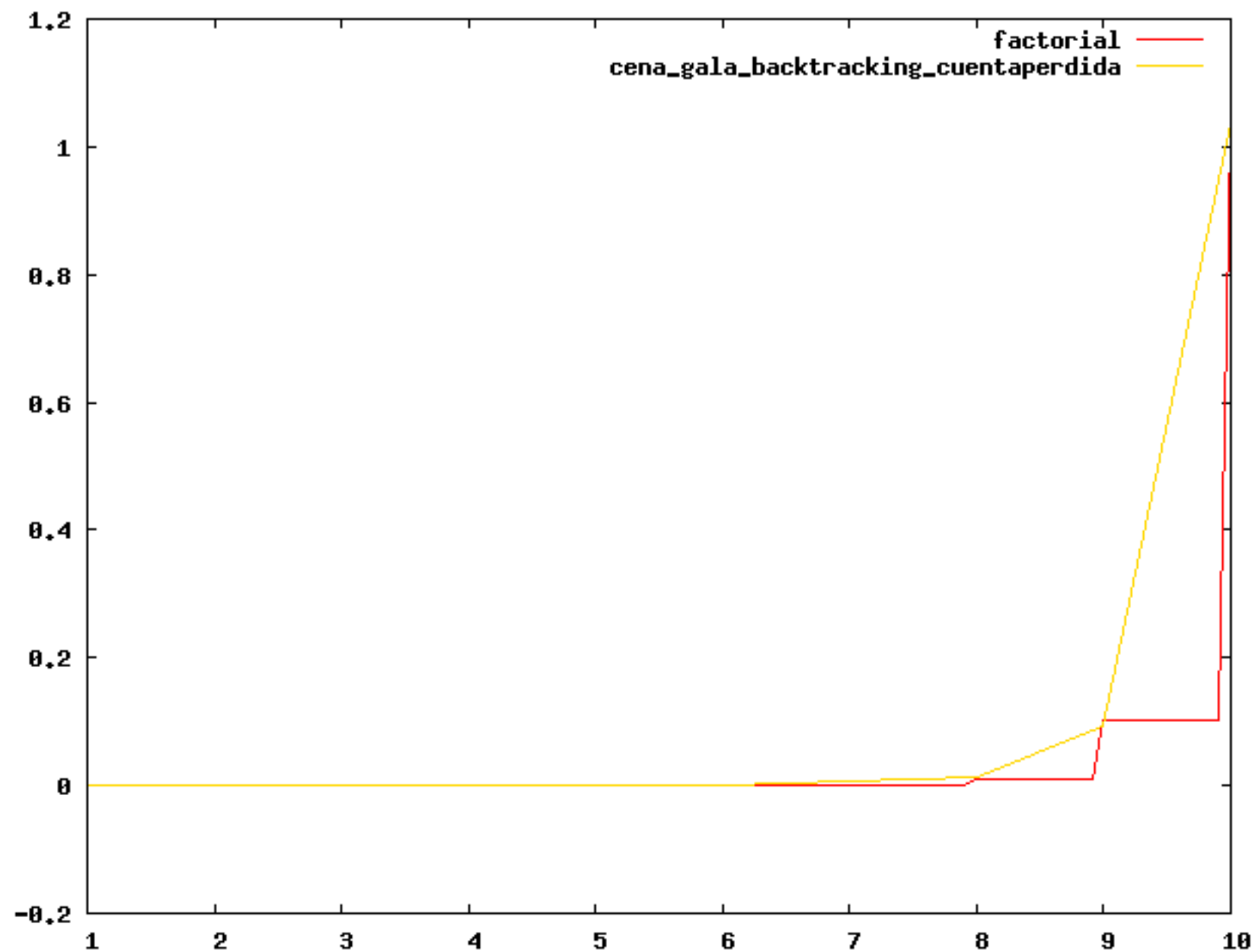
```
//Cuenta la afinidad que ya has perdido cogiendo la distribución por ahora hecha en rama
int afinidadNoConseguida(list<int> rama, int* afinidades, int n){
    int res = 0;
    list<int>::iterator itnext ;
    for(list<int>::iterator it = rama.begin(); it != prev(rama.end()); it++){
        itnext = it;
        itnext++;
        res += afinidades[(it)*n+(*(itnext))];
    }
    res += afinidades[(*(rama.begin()))*n+(*(prev(rama.end())))];
    return 100*rama.size() - res;
}
```

-uuu:**-F1 cena_gala_backtracking_cuentaperdida.cpp 6% L11 (C++/l Abbrev)-----
Auto-saving...done

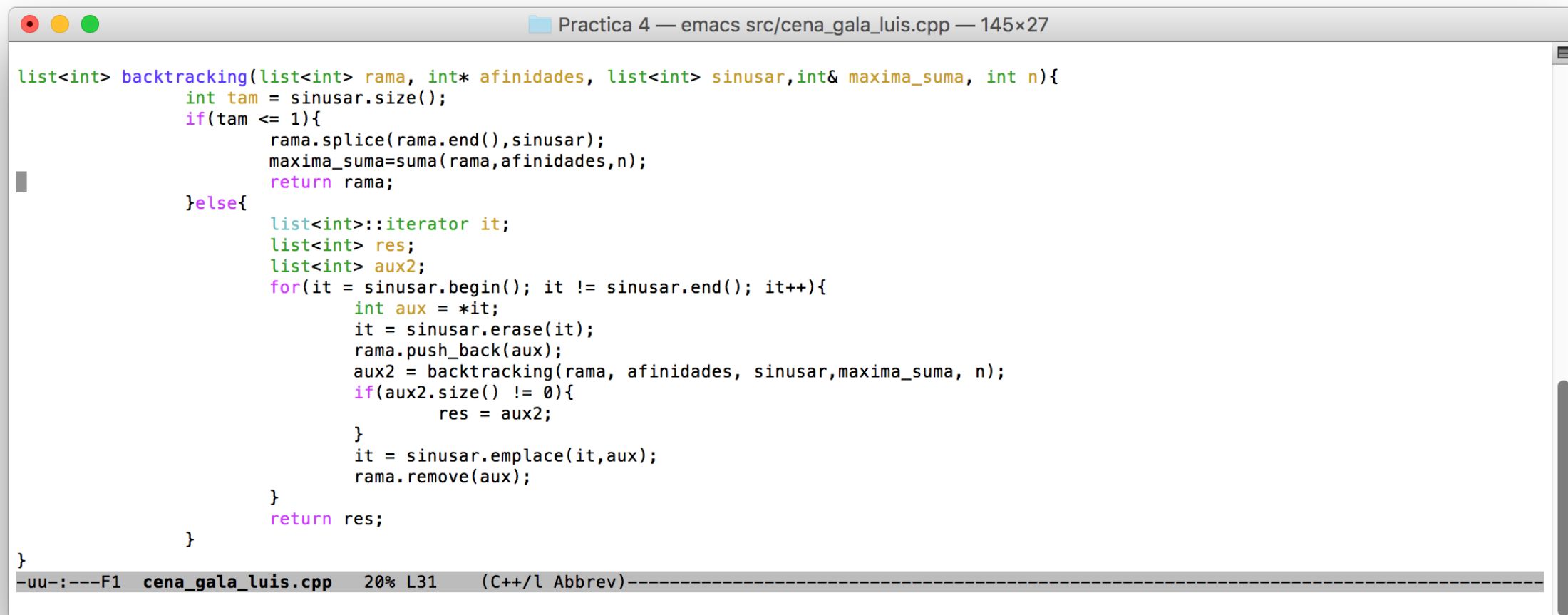
CENA DE GALA BACKTRACKING CUENTA PERDIDA – (ALGORITMO FACTORIAL)

	Mac	Fujitsu	Toshiba I	Toshiba II
1	5.62e-06	2.15e-06	1.623e-06	8.553e-06
2	5.226e-06	3.044e-06	2.76e-06	1.325e-06
3	1.4487e-5	8.35e-06	1.032e-05	2.565e-05
10	0.0129116	0.087247	0.0237846	0.0062321
11	0.0922949	1.39745	0.14221	0.115942
12	1.03912	95.692	1.35433	1.47772

CENA DE GALA BACKTRACKING CUENTA PERDIDA – (ALGORITMO FACTORIAL)



CENA DE GALA LUIS- (ALGORITMO FACTORIAL)

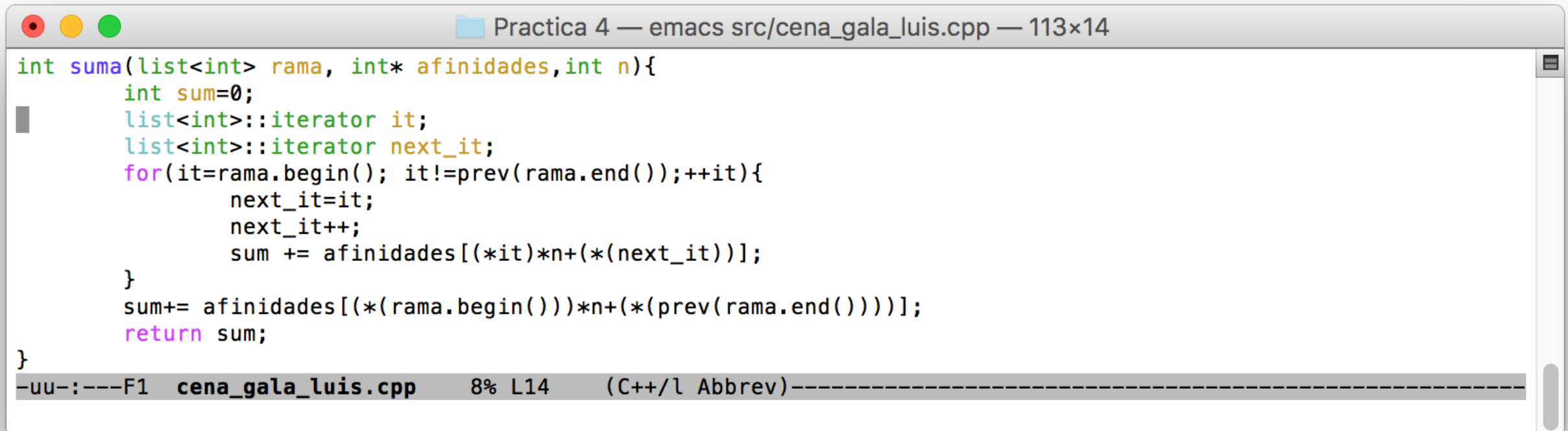


The image shows a screenshot of an Emacs editor window. The title bar at the top reads "Practica 4 — emacs src/cena_gala_luis.cpp — 145x27". The code is written in C++ and implements a backtracking algorithm. The code is as follows:

```
list<int> backtracking(list<int> rama, int* afinidades, list<int> sinuser, int& maxima_suma, int n){
    int tam = sinuser.size();
    if(tam <= 1){
        rama.splice(rama.end(), sinuser);
        maxima_suma=suma(rama,afinidades,n);
        return rama;
    }else{
        list<int>::iterator it;
        list<int> res;
        list<int> aux2;
        for(it = sinuser.begin(); it != sinuser.end(); it++){
            int aux = *it;
            it = sinuser.erase(it);
            rama.push_back(aux);
            aux2 = backtracking(rama, afinidades, sinuser,maxima_suma, n);
            if(aux2.size() != 0){
                res = aux2;
            }
            it = sinuser.emplace(it,aux);
            rama.remove(aux);
        }
        return res;
    }
}
```

The status bar at the bottom of the Emacs window shows: "-uu-:---F1 cena_gala_luis.cpp 20% L31 (C++/l Abbrev)-----".

CENA DE GALA LUIS- (ALGORITMO FACTORIAL)

A screenshot of an Emacs editor window. The title bar at the top reads "Practica 4 — emacs src/cena_gala_luis.cpp — 113x14". The code is written in C++ and uses syntax highlighting. It defines a function named 'suma' that takes a list of integers 'rama', an array of integers 'afinidades', and an integer 'n'. The function calculates a sum based on the elements of 'rama' and 'afinidades'. The status bar at the bottom shows "-uu-:---F1 cena_gala_luis.cpp 8% L14 (C++/l Abbrev)-----".

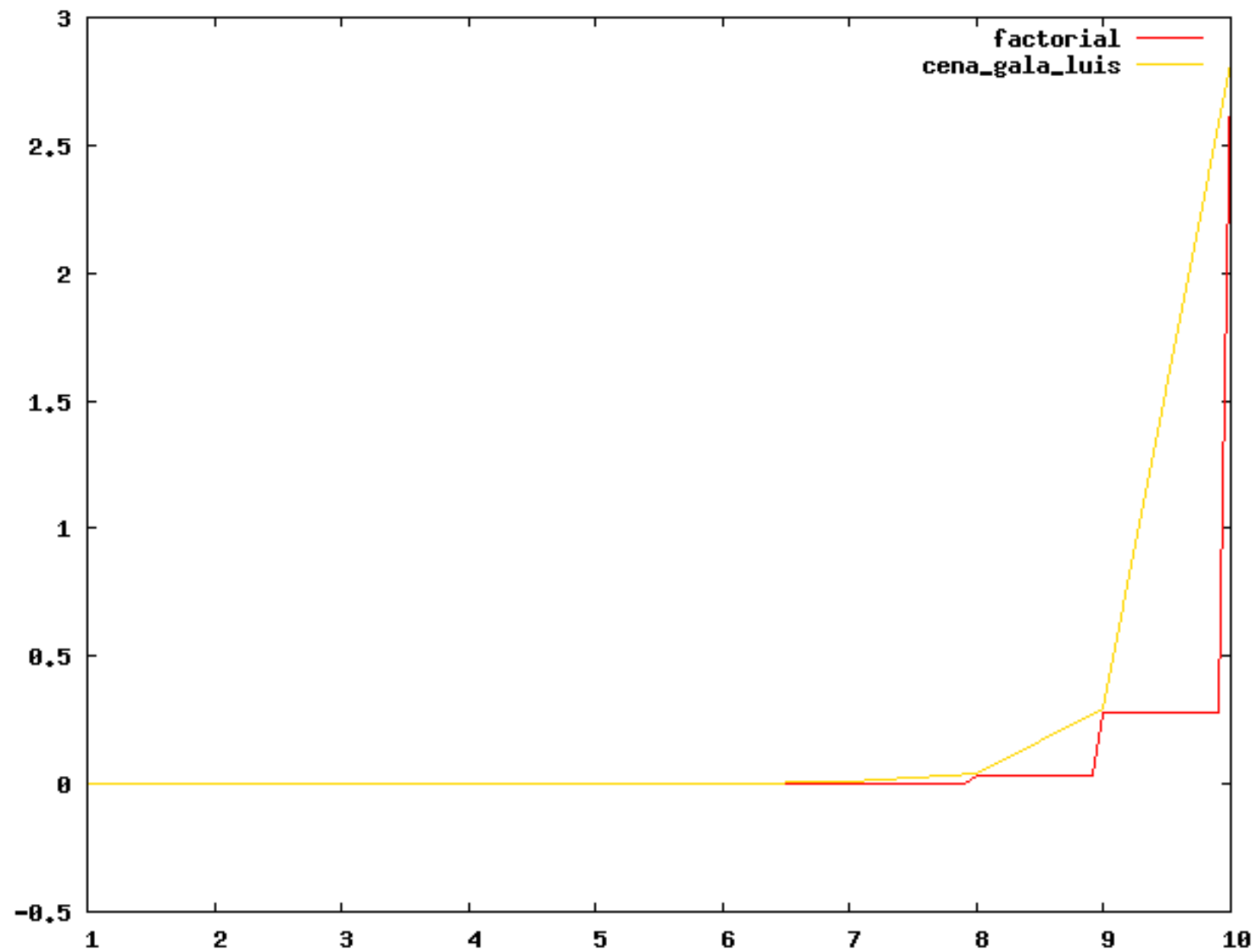
```
int suma(list<int> rama, int* afinidades, int n){
    int sum=0;
    list<int>::iterator it;
    list<int>::iterator next_it;
    for(it=rama.begin(); it!=prev(rama.end());++it){
        next_it=it;
        next_it++;
        sum += afinidades[(*(it)*n+(*(next_it)))];
    }
    sum+= afinidades[(*(rama.begin())*n+(*(prev(rama.end()))))];
    return sum;
}
```

-uu-:---F1 cena_gala_luis.cpp 8% L14 (C++/l Abbrev)-----

CENA DE GALA LUIS – (ALGORITMO FACTORIAL)

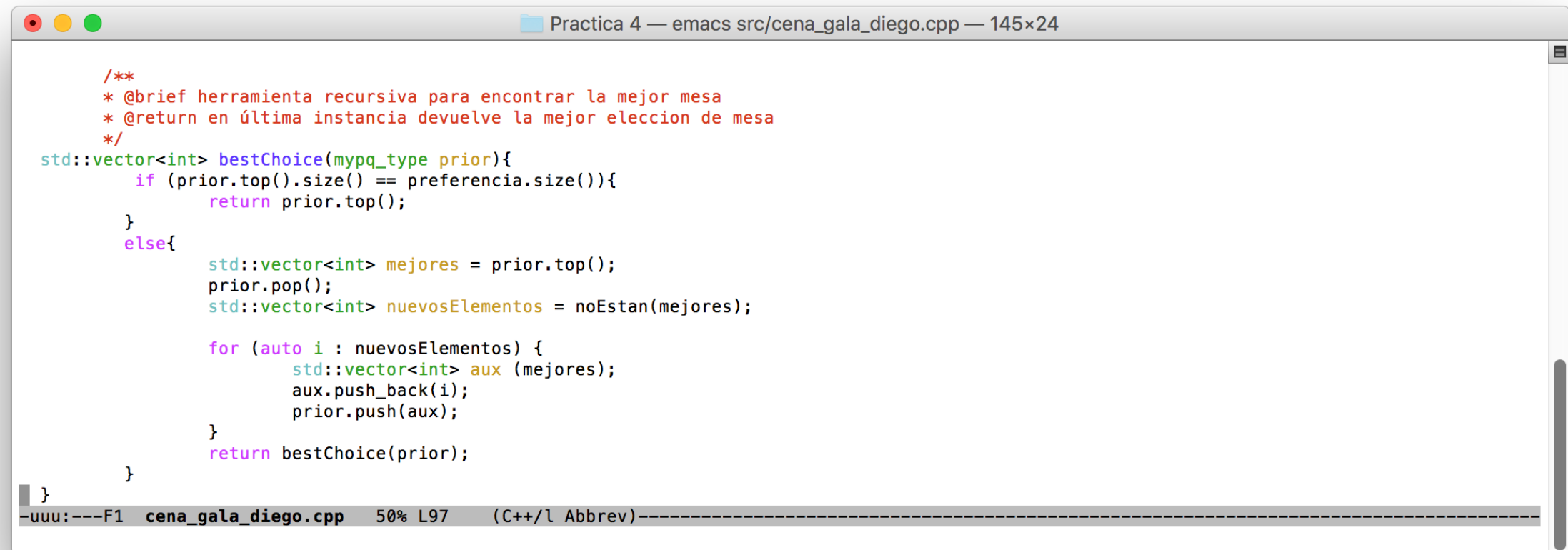
	Mac	Fujitsu	Toshiba I	Toshiba II
1	5.156e-06	3.757e-06	3.238e-06	8.125e-06
2	4.823e-06	4.954e-06	2.305e-06	6.842e-06
3	1.3695e-5	1.388e-05	6.519e-06	2.736e-05
8	0.03977719	0.0257423	0.0205837	0.03699937
9	0.294831	0.182338	0.169754	0.125047
10	2.82424	1.8034	1.68421	1.42412

CENA DE GALA LUIS- (ALGORITMO FACTORIAL)



BRANCH AND BOUND

CENA DE GALA BRANCH N BOUND – (ALGORITMO FACTORIAL)



The image shows a screenshot of an Emacs editor window. The title bar at the top reads "Practica 4 — emacs src/cena_gala_diego.cpp — 145x24". The code is written in C++ and implements a recursive function named `bestChoice`. The code is as follows:

```
/**
 * @brief herramienta recursiva para encontrar la mejor mesa
 * @return en última instancia devuelve la mejor eleccion de mesa
 */
std::vector<int> bestChoice(mypq_type prior){
    if (prior.top().size() == preferencia.size()){
        return prior.top();
    }
    else{
        std::vector<int> mejores = prior.top();
        prior.pop();
        std::vector<int> nuevosElementos = noEstan(mejores);

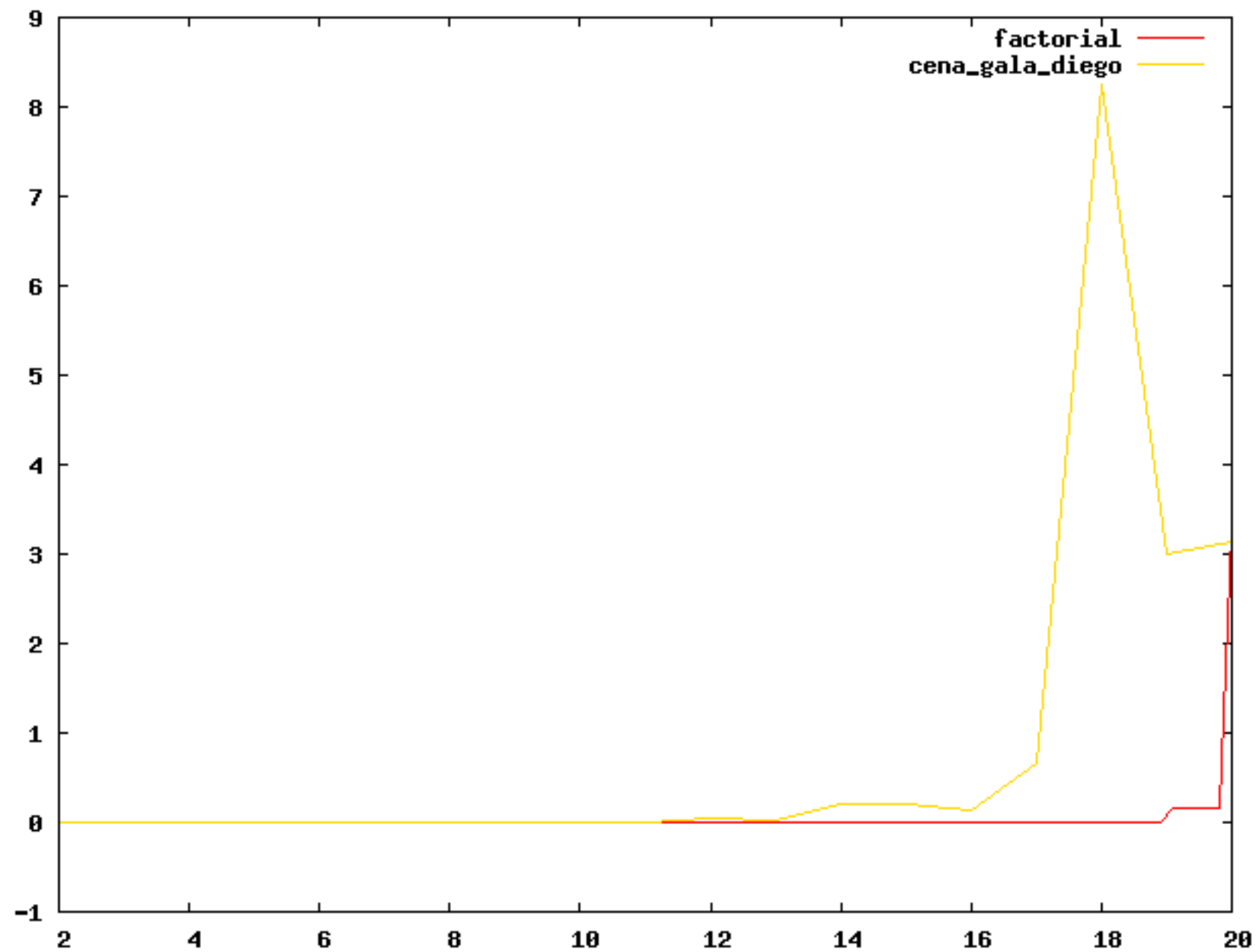
        for (auto i : nuevosElementos) {
            std::vector<int> aux (mejores);
            aux.push_back(i);
            prior.push(aux);
        }
        return bestChoice(prior);
    }
}
```

The status bar at the bottom of the window displays: `uuu:---F1 cena_gala_diego.cpp 50% L97 (C++/l Abbrev)`.

CENA DE GALA BRANCH N BOUND – (ALGORITMO FACTORIAL)

	Mac	Fujitsu	Toshiba I	Toshiba II
2	0.0004949	2.15e-06	1.623e-06	0.0728696
3	2.408e-05	3.044e-06	2.76e-06	0.293622
4	8.013e-05	8.35e-06	1.032e-05	0.654133
18	8.24175	0.087247	0.0237846	1.1646
19	2.99178	1.39745	0.14221	1.80405
20	3.12603	95.692	1.35433	2.61626

CENA DE GALA BRANCH N BOUND – (ALGORITMO FACTORIAL)



**OTROS
ALGORITMOS**

CENA DE GALA NACHO – (ALGORITMO CUADRÁTICO)

```
Practica 4 — emacs src/cena_gala_nacho.cpp — 109x52

void ColocaComensales(short int** matriz, int n_comensales, list<int>& resultado_final)
{
    //Recorro la lista de resultados para ver el mejor lugar para insertar el siguiente comensal.
    //El índice i es el comensal actual y el j es el que recorre la lista resultado_final.
    for(int i = 0; i < n_comensales; ++i)
    {
        list<int>::iterator mejor_posicion = resultado_final.begin();
        int mejor_afinidad = 0;

        for(list<int>::iterator iter = resultado_final.begin(); iter != resultado_final.end(); ++iter)
        {
            //Hallo el iterador de antes de end.
            list<int>::iterator antes_end = resultado_final.begin();
            for(list<int>::iterator iter2 = resultado_final.begin(); iter2 != resultado_final.end(); ++iter2)
            {
                list<int>::iterator aux = iter2;
                ++aux;
                if(aux != resultado_final.end())
                    antes_end = aux;
            }

            //Asigno los pesos de cada comensal para sus acompañantes a izquierdas y derechas.
            //La lista la considero circular, el primero esta al lado del último y viceversa.
            int afinidad_izq = 0, afinidad_der = 0;
            if(iter == resultado_final.begin() || iter == antes_end)
            {
                list<int>::iterator aux = iter;
                ++aux;
                afinidad_izq = matriz[i][*iter];
                afinidad_der = (aux != resultado_final.end()) ? matriz[i][*aux] : matriz[i][resultado_final.front()];
            }
            else
            {
                list<int>::iterator aux = iter;
                ++aux;
                afinidad_izq = matriz[i][*iter];
                afinidad_der = matriz[i][*aux];
            }

            //Cambio mejor afinidad y mejor posicion si he encontrado un lugar mejor donde insertar;
            int puntuacion = afinidad_der + afinidad_izq;
            if(puntuacion > mejor_afinidad)
            {
                mejor_posicion = iter;
                mejor_afinidad = puntuacion;
            }
            if(puntuacion == 200)
                break;
        }
    }
}
```

---uuu:---F1 cena_gala_nacho.cpp 4% L17 (C++/l Abbrev)-----

CENA DE GALA NACHO – (ALGORITMO CUADRÁTICO)

	Mac	Fujitsu	Toshiba I	Toshiba II
50	0.0007452	0.0013751	0.0007401	0.0007137
51	0.0007971	0.0020129	0.0008154	0.0007415
52	0.000844	0.0016592	0.0009061	0.0007907
248	0.0842584	0.141345	0.0916732	0.0812439
249	0.0865886	0.171309	0.093271	0.0812332
250	0.092747	0.107369	0.0949944	0.0826273

CENA DE GALA NACHO – (ALGORITMO CUADRÁTICO)

