Subject: Re: Map implementation
Posted by mirek on Tue, 16 Apr 2019 10:38:40 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 14 April 2019 18:55
Added 100 elements
Mem used: 2592740 Kb
[/code]

IMHO, it is possible to do much better than this ...

Just for reference (Visual C++ 64 bit release):

```
#include <Core/Core.h>
#include <set>
#include <vector>

using namespace Upp;

CONSOLE_APP_MAIN
{
 int curMU = MemoryUsedKb();
 int v_num = 1000000;
 std::vector< std::set<int> > v;
 Cout() << "sizeof(Index<int>): " << sizeof(Index<int>) << " bytes" << EOL;
 Cout() << "Mem used: " << MemoryUsedKb() - curMU << " Kb" << EOL;
 v.resize(v_num);
 Cout() << "Created " << v_num << " empty Index<int>" << EOL;
 Cout() << "Mem used: " << MemoryUsedKb() - curMU << " Kb" << EOL;
 const int isize = 100;
 for (int i = 0; i < isize; ++i) {
  const int jsize = v_num;
  for (int j = 0; j < jsize; ++j)
   v[j].insert(i);
  Cout() << "Added " << i + 1 << " elements" << EOL;
  Cout() << "Mem used: " << MemoryUsedKb() - curMU << " Kb" << EOL;
 }
}
```

Added 100 elements
Mem used: 3222476 Kb

Replace set with unorderd_set and

Added 100 elements
Mem used: 12412392 Kb

sizeof(std::unordered_set<int>) = 64

GCC 64bits, unorderd_set

Added 100 elements
Mem used: 4621148 Kb

sizeof(std::unordered_set<int>) = 56

BTW, I am now banging my head about reducing overhead from 20 bytes to 16 per element. It looks like it might not be possible without performance degradation (in all scenarios). So I would say it is really hard to get below 20 per element.

Mirek